Proceedings in MANUFACTURING SYSTEMS

Proceedings in Manufacturing Systems, Volume 14, Issue 1, 2019, 9-14 ISSN 2067-9238

TASK-ORIENTED OFF-LINE PARAMETRIC PROGRAMMING OF AN INDUSTRIAL ROBOT SERVING CNC LATHES

George VOSNIAKOS[1,*], Athena PAROUSI[2], George LEIVADIOTAKIS[3], Lucas BLEIN[4], Charlotte LASSIME[5]

[1)] Prof., Manufacturing Technology Section, School of Mechanical Engineering, National Technical University of Athens, Greece
[2)] Dipl. Eng., Manufacturing Technology Section, School of Mechanical Engineering, National Technical University of Athens, Greece
[3)] M. Eng., Manufacturing Technology Section, School of Mechanical Engineering, National Technical University of Athens, Greece
[4)] Dipl. Eng., Ecole National d' Ingenieurs de St Etienne (ENISE), Saint-Etienne, France
[5)] Dipl. Eng., Ecole National d' Ingenieurs de St Etienne (ENISE), Saint-Etienne, France

Abstract: *Industrial robotic arms are often employed in machine tool tending. If the scope of programming is restricted to a specific machine tool type, the tasks involved become standard to a large extent. Therefore, the corresponding movements can be parametrically described and linked to the shape of the particular machine tool tended. Thus, a 'master' robot trajectory and corresponding robot program in the supported language can be parametrically defined off-line by making use of a simplified 3D model of the machine tool and an accurate 3D model of the robot. Subsequently, this can be easily tailored to the particular machine tool of the given type by simply updating the respective parameter values. This process has been implemented and demonstrated for a six axis industrial robot tending computer numerically controlled lathes.*

Key words: *robot programming, parametric, tasks, off-line, simulation.*

1. INTRODUCTION

Industrial robot programming has been being approached in various novel ways recently, notably: teaching the robot its path by demonstration referring to human gestures or metaphors on real or on virtual environments [1], using augmented reality to provide complementary information to the human programmer, [2] etc. However, industrial robots are mostly still being programmed by CAD-based off-line method, essentially kinematic simulation, or alternatively by lead-through on-line method. The reason is that these are well tested over the years and, of course, that there are a large number of robotic arms of the previous generation(s) still operational in the manufacturing industry.

Off-line programming can be better accomplished if the robot's path is broken down into segments, each being associated with a particular task. Both tasks and segments can be defined parametrically, thus allowing for easy adaptability to different robots, collaborating machinery and production environment in general.

Task-based robot programming started being exploited two decades ago, yet not in its fully parametric version, which was made possible with the establishment of constraint based kinematic solvers one decade later.

The first approaches pertaining to task-based programming of industrial robots either proposed a programming language describing the production environment and a flow of activities [3] or decomposed the programming problem into several layers, automatically mapped to programming instructions which were tested on a simulator of the full robotic cell [4]. Since tasks were especially evident to define in the case of assembly tasks, an assembly language not only described these tasks but associated robot movements with them by referring to CAD-based kinematic modelling [5]. A complex task needed to be broken down into sub-tasks, easy enough for straightforward mapping into robot movements [6].

Off-line parametric programming based on forward kinematics or simultaneous movement of up to 3 joints relies on accurate 3D models of the robot, jigs and cell equipment and collision detection capability, e.g. in press-brake tending [7]. Speed and accuracy of end-effector centered programming are enhanced by constraint-based modelling, as demonstrated in the case of welding tasks [8], machine tool loading/ unloading [9] and, generally, paths resulting from complex tasks [10].

Robotic tending of machine tools reduces non-productive time spent in part handling and makes small batches economically viable. Thus, fast programming of the robot becomes a necessity. In this work, development of a parametric application of loading / unloading lathes using a particular robotic arm is presented, demonstrating ease and flexibility of constructing robot programs for any part that needs to be processed on CNC lathes.

Section 2 presents the type of machine tools targeted, the robot and the developed end effector. Section 3 deals with generic task definition and parametrisation. Section 4 outlines the link to robot program commands. Section 5 presents a typical case study and Section 6 summarises the findings overall.

[*] Corresponding author: Heroon Politehniou 9, 15780 Athens, Greece
Tel.: +302107721457; Fax: +302107724273
E-mail addresses: *vosniak@central.ntua.gr* (G. Vosniakos), *athinadasx @hotmail.com* (A. Parousi), *gleivadiwtakis@yahoo.gr* (G. Leivadiotakis), *lucas.blein @enise.fr* (L. Blein), *charlotte.lassime@enise.fr* (C.Lassime)

2. EQUIPMENT

2.1. CNC lathes

CNC lathes are the targeted machine tools that need to be loaded and unloaded by the robot. They are modelled in CAD environment in a simplistic way, focusing only on their parts and dimensions that matter, i.e. the spindle axis, the chuck, quill and the available space behind the door and within covers, see Fig. 1.

2.2. Robot

A six link Stäubli RX90L robot is used as a typical example of industrial robotic arms that can be programmed following the advocated method. It weighs 112 kg and has a payload of 6 kg at nominal speed. Its configuration is given in Fig. 2. Its joint speed and range data are given in Table 1.

2.3. End effector

The end effector, a pneumatic two-finger gripper, was designed and manufactured in-house so as to enable secure holding of axisymmetric parts that can fit into a cube of side 20 to 150 mm. The actuator is a double action cylinder of 50 mm stroke, 10 bar max pressure, 458 / 394 N indicative push/pull force (at 6 bar). The gripper consists of 28 parts of Aluminium alloy series 2000 weighing 3.7 kg, see Fig. 3.

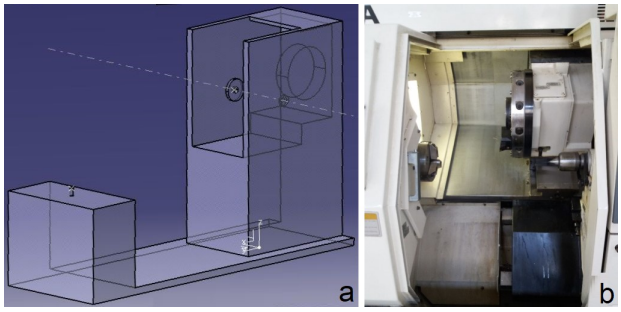


Fig. 1. CNC lathe: *a* – simplified model; *b* – real counterpart.

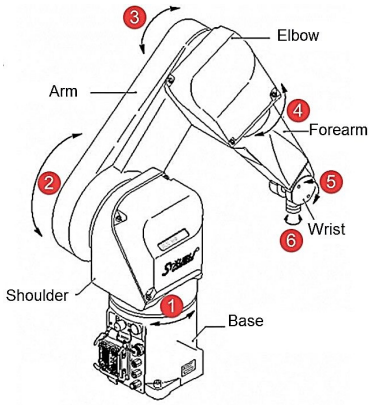


Fig. 2. Robot configuration.

Table 1

Robot data

Joint (n)	1	2	3	4	5	6
Working range(°)	±160	±137.5	±142.5	±270	+120-105	±270
Nominal speed (°/s)	236	200	286	401	320	580
Max speed (°/s)	356	356	296	409	480	1125
Distance to J_{n-1} (mm)	0	420	450	0	650	85

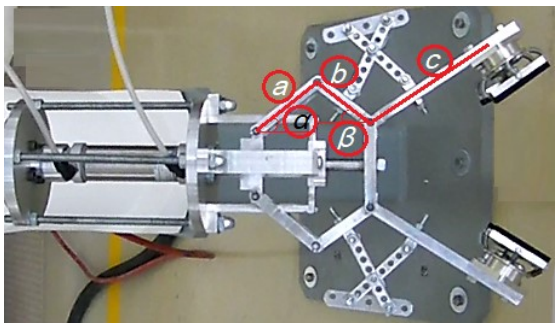


Fig. 3. Pneumatic gripper developed.

The maximum weight of the handled part is 6 − 3.7 = 2.3 kg. A substantial feature of the gripper is the double plate finger, each plate being spherically hinged with maximum deviation ± 20° and spring loaded for automatic return to equilibrium. This design enables adaptability to different curvatures of the handled object.

The force exerted by each finger normal to the grabbing surface F is [11]:

$$F=P \sin(\alpha+\beta)\, b / (2\, c \sin \alpha), \quad (1)$$

where P is the force exerted by the pneumatic cylinder, a, b, c are the lengths of the linkages and α, β are angles characterising the pose of the mechanism, see Fig. 3. Gripping force is calculated at 23 and 33 N for stroke 1 and 39 mm, respectively, at 4 bar pressure corresponding to P = 120 N. A pressure increase up to 10 bar increases the gripping force accordingly. Using finite element analysis (Solidworks Simulation™) maximum stress was found to be much lower than yield stress, see Fig. 4.

The maximum velocity of movement for the handled part was calculated by considering the friction force exerted through the rubber pads of the fingers with a coefficient of friction μ = 0.7 and the inertial forces resulting from a maximum acceleration of 2g and possibly centrifugal forces, at a radius 290 to 1100 mm, for different cases, see Fig. 5, as well as mass handled.

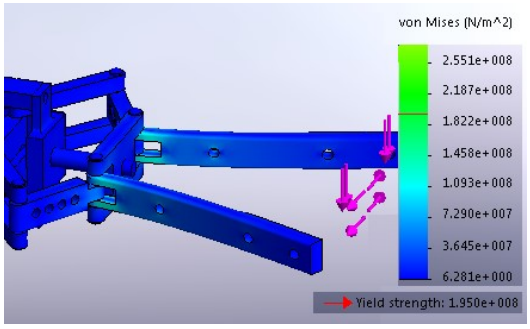


Fig. 4. Strength analysis of the gripper.

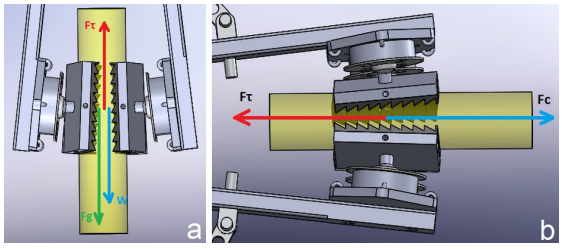


Fig. 5. Limit loading cases: *a* – lifting; *b* – centrifuge.

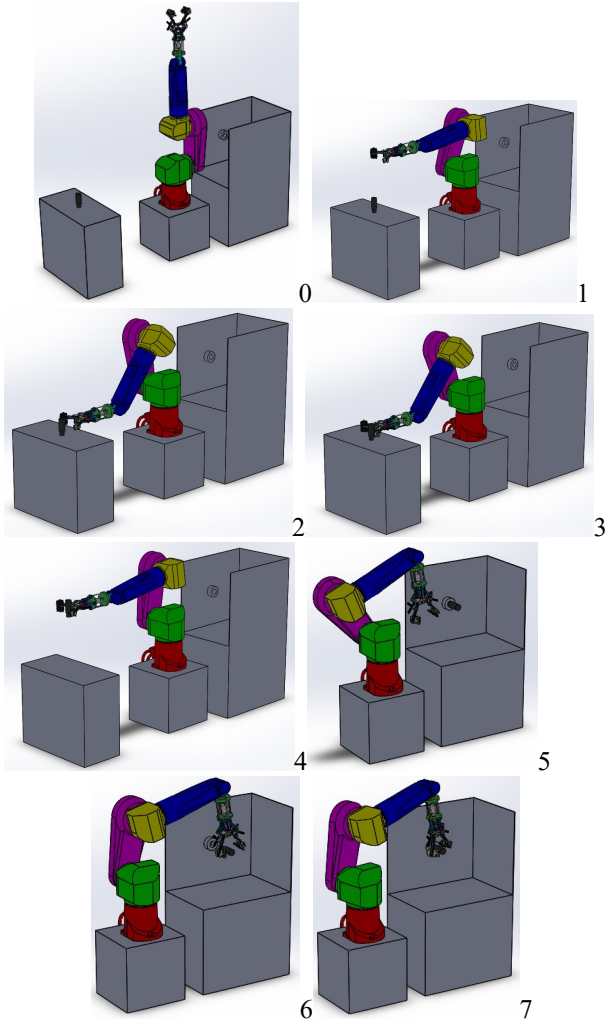


Fig. 6. Characteristic points (robot poses) in loading operation.

Table 2

Definition of lathe loading (CP/NP: current/next pose, I: interpolation, S: straight, H: horizontal, V: vertical)

No	CP	NP	Motion type	Gripper mode	Sub Task	Task
0	0	0	-	Open	Dwell	Wait empty
1	0	1	I-H	Open	Approximate horizontal offset align from home	Move to pick up part
2	1	2	S-V	Open	Accurate down vertical align	
3	2	3	I	Open	Accurate horizontal align	
4	3	3	-	Closed	Grip	Grip part
5	3	4	S-V	Closed	Accurate up vertical align	Move part to home
6	4	1	S-H	Closed	Accurate horizontal move	
7	1	0	I-H	Closed	Horizontal homing	
8	0	5	I	Closed	Lathe near	Load part to chuck
9	5	6	S	Closed	Enter lathe & align with chuck axis	
10	6	7	S-H	Closed	Insert into chuck	
11	7	7	-	Open	Open gripper	Leave part
12	7	6	S-H	Open	Axial retract from chuck	Leave lathe workspace
13	6	5	S	Open	Radial exit from lathe	
14	5	0	I	Open	Return to home	

Overall, the maximum attainable velocity of 12 m/sec is allowable only for handled part mass up to 0.5 kg, dropping to 4 m/sec for maximum allowable mass of 2.3 kg.

3. LOADING / UNLOADING TASK DEFINITION

The loading job can be broken down into tasks, each of which is further broken down into sub-tasks. Sub-tasks are defined in such a way as to enable easy adaptation to any lathe and part dimensions. This largely means that each sub-task corresponds to a collision-less movement. Sub-tasks make use of a home position and another 7 positions (poses). Note that the term position refers to the position and orientation of the end effector (gripper) is 3D space as well as the robot pose. The most important of these positions are shown in Fig. 6, whereas the tasks, sub-tasks and corresponding positions are summarized in Table 2. The unloading job is defined in a very similar way to loading, except in the inverse direction.

Each robot position is defined parametrically with respect to the main dimensions of the lathe, the robot and the part at hand, see Table 3.

There are two methods to calculate the robot joint positions corresponding to characteristic points (0)-(7), namely analytic geometry calculations in closed form and CAD-based kinematics.

3.1 Analytic geometry calculations

Closed analytic solutions can be found for joint coordinates for the tasks given in Table 2. As a first example, the case referring to the end effector's entry into the lathe's workspace, see Fig. 7, is indicatively solved next.

Note that all linear dimensions depicted in Fig. 7 are either input as parameters of the robotic cell, see Table 3, or directly derived from them. The angles depicted are solved for by first fixing gripper orientation at $v = 80^{o}$. Then, Eq. (2) and (3) can be written and can be combined to yield Eq. (4):

$$d_e = B \cos w + C \cos v - A \sin \alpha, \quad (2)$$

$$z_e = H + A \cos \alpha + B \sin w - C \sin v, \quad (3)$$

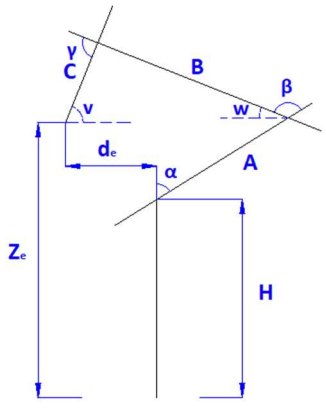
$$B^2 = (d_e - C \cos v)^2 + (z_e - H + C \sin v)^2 + A^2 + 2A\,[(d_e - C \cos v) \sin \alpha - (z_e - H + C \sin v) \cos \alpha]. \quad (4)$$

Table 3

Robotic cell dimension parameters

	Dimension	Set1	Set2
Lathe	Axis height (h_c)	992	1100
	Upper free point height	1340	1500
	Free axis length	415	620
	Workspace depth	400	490
	Chucked part length	30	30
Robot	Length at home point	1605	1605
	Robot base height	500	500
	Gripper length	455	455
	Contact length with part	100	100
Part	Length	138	180
	Storage base height	787	787
	Gripping point -unprocessed	83	125
	Machined length	138	200
	Gripping point -processed	83	125


a



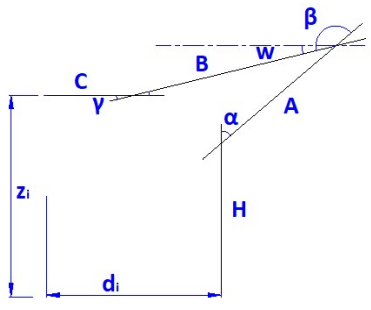
b

Fig. 7. Pose 5: *a* – real robot; *b* – calculation sketch.

Equation (4) is solved for angle α, and, then, Eq. (2) is solved for angle *w*. Then, angle $\beta = w - \alpha - \pi/2$ and $\gamma = v + w$. Referring to Fig. 7, angle α is positive and angles β, γ are negative.

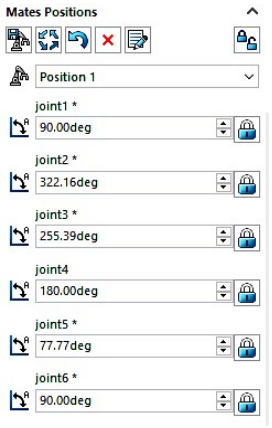
As a second example, the case referring to approximate alignment of the gripper above the part to be gripped, see Fig. 8, is solved for joint angles α, β, and γ.

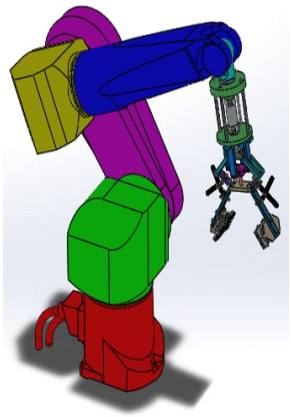

a



b

Fig. 8. Pose 1: *a* – real robot; *b* – calculation sketch.





a b

Fig. 8. Simulation in Solidworks™: *a* – mate control; *b* – pose.

$$d_i = B \cos w - A \sin \alpha + C, \quad (5)$$

$$z_i = H + A \sin \alpha - B \cos w. \quad (6)$$

Equations (5) and (6) yield:

$$B^2 = (d_i - C)^2 + (H - z_i)^2 + A^2 + 2A[(H - z_e) \cos \alpha + (d_i - C) \sin \alpha]. \quad (7)$$

Equation (7) is solved for α (Joint 2). Then, Eq. (5) yields *w*, which equals angle γ (Joint 5) and since $w = 90 + \alpha - \beta$ angle β results (Joint 3). Angles α, γ are positive, whilst β is negative.

All data entered is checked for validity as follows:

(a) All joints' coordinates (angles) need to be within the range defined in the robot's specifications. If they are not, the robot's relative position to the lathe needs to be changed.

(b) The part's length has to be able to fit in the free length of the lathe's workspace

(c) The lathe's axis needs to be parallel to the robot's global x-axis, otherwise their angle needs to be measured with accuracy and be used to correct the calculations.

The calculated joint coordinates corresponding to each of the 8 poses of the robot are stored in a spreadsheet. They are used to define the respective movements, either as straight or interpolated segments as stated in Table 2.

The movements are simulated using the kinematics mode of Solidworks™. The model of the robot is made available by the manufacturer, as is the case with most robots nowadays. Key to the kinematics simulation is the definition of angle 'mates'. The mate controller menu enables the user to check the respective pose, essentially in forward kinematics mode, transforming angles in the range 0°–360°, see Fig. 8.

It is also possible to compose a single animation file out of these consecutive poses, thus obtaining a continuous flow of robot's movements.

3.2 CAD-based kinematics

If analytic calculations for points 0 to 7 as outlined in section 3.1 are not desired, a lead-through approach can be followed using the digital model of the robot. Analytic calculations are based on moving one joint at a time sequentially to control the path closer. Of course, the same points could be used for multiple joint simultaneous movement, but in that case the exact path

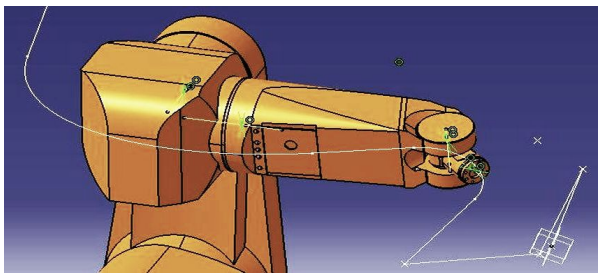


Fig. 9. CAD-based detailed path planning.

would not be known, which might entail collision risks. Furthermore, accurate path description in 3D space is practically impossible outside a CAD-based environment. Such description will keep characteristic points 0–7 but would add further points to modify the path, for instance, to better control object clearance, see Fig. 9. Apart from the fact that the resulting path description is richer, the procedure is the same, i.e. point coordinates will also be stored in a spreadsheet from which the respective values are filled in into the robot program subsequently. As in the previous case, see Section 3.1, kinematic simulation is useful for collision detection, too.

4. ROBOT PROGRAM DERIVATION

The robot program is essentially standard in terms of the flow of commands as well as the very commands as such, because the tasks and sub-tasks are standard, the differences pertaining to numerical values of coordinates.

These are read from specific cells known beforehand in the spreadsheet where the point coordinates are stored after either analytic or CAD-based calculations.

The programming language is V+ [12], but this is very similar to other languages, especially those descending from the VAL family. The following commands are made use of:

(a) *DRIVE joint, angle change, % of max speed*: operates one joint in forward kinematics mode. This is mainly used for approaching to and departing from points 4 & 5.

(b) *MOVES point*: simultaneously activates any necessary joints to achieve interpolated straight line movement of the gripper**.** This is mainly used for points 5, 6 and 7, i.e. movement inside lathe workspace in inverse kinematics.

(c) *MOVE*: simultaneously activates any necessary joints to achieve interpolated movement of the gripper. This is mainly used when precise path control is not needed.

(d) *SET*: Defines a pose in forward or inverse kinematics.

(e) *BREAK*: This stops execution of the next commands until completion of the current one.

(f) *DELAY*: This introduces dwell of prescribed duration. It is used in waiting for machining and opening / closing of the chuck.

(g) *OPENI – CLOSEI*: The gripper is open / closed before execution of the next command starts.

Any robot pose is defined in the following ways:

(a) *#PPOINT***:** This prescribes six joint coordinates. It is used for defining point 0.

(b) *SHIFT (transformation BY x_s, y_s, z_s):* This shifts an initial point along z, y, z global axes. For instance, it is used for defining points at the end of straight line movements to and along chuck axis.

(c) *HERE location_var*: This defines a location by assigning it the current joint coordinates. It is exploited in order to define reference points that have been reached in forward kinematics mode and need to be used subsequently in inverse kinematics mode.

(d) *TRANS (x,y,z,y,p,r)*: This defines a pose in inverse kinematics mode, i.e. location (x,y,z) and orientation (yaw, pitch, roll) of the end effector. It is used when a closely controlled path is needed, defined by consecutive poses, instead of a freely interpolated path based on its start and end.

There are also two logical variables to check whether the chuck is open / closed and whether the lathe's door is open / closed. Their values may be assigned by the lathe's controller when direct connection is possible, otherwise they may be assigned by the user interactively.

An example of usage of the above commands in the V+ program follows, parameter values being underlined:

```
SET #00 = #PPOINT (1,-89,91,1,1,1)          ; pose 0
MOVE #00
BREAK
…
SET P3=TRANS(32.52,991.14,158.68,0,180,-90);pose3
MOVE P3
…
DRIVE 2,-39.56,80
BREAK
DRIVE 5,103.23,80
BREAK
DRIVE 3,106.33,80
BREAK
SPEED ALWAYS,20
HERE P5                                     ;pose 5
SET P6 = SHIFT (P5 BY 0,400,0)
MOVES P6                                    ;pose 6
BREAK
SET P7 = SHIFT (P6 BY -143.41,0,0)
MOVES P7                                    ;pose 7
DELAY 10
OPENI                                       ;leave part
```

5. RESULTS

The approach described was implemented for two different CNC lathes, an OKUMA LB10II and a HAAS TL-1. The dimensions of the part were obtained by direct measurement, whereas those of the lathes were obtained by the respective specifications.

A special issue pertains to the relative position and orientation of the robot with respect to the lathe. Photogrammetry was used to measure the exact distance of the robot base to the chuck axis as well as the latter's inclination angle to the robot's x-axis, see Fig. 10(c).

Imetric™'s photogrammetry system was used, employing coded and non-coded targets, a high-resolution camera (a Nikon™ D90 with a Sigma™ electronic flash), calibration scale bars, see Fig. 10(a-b), as well as Imetric™ proprietary software for calculating the coordinates of the targets in 3D space and fitting primitive shapes to them, see Fig. 10(d).

Eight different results were obtained and their mean was extracted as: $\theta = 2.07^{o}$, $\alpha = 812.16$ mm, $b = 197.46$ mm, $h_c = 992.51$ mm.

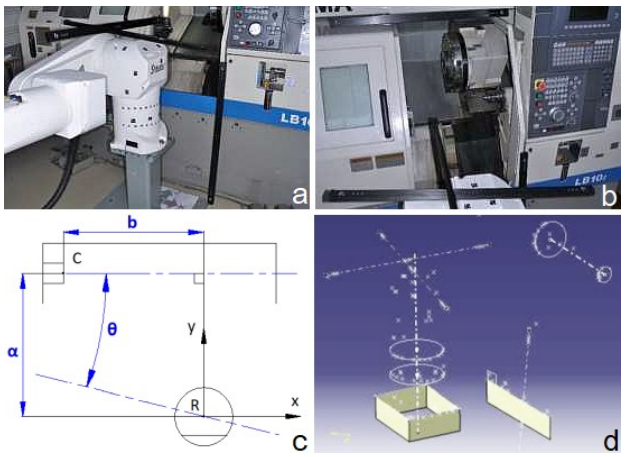


Fig. 10. Photogrammetry measurement: *a* and *b* – setup; *c* – plan view of target layout; *d* – results output to CAD system.

Fig. 11. Application example of CNC lathe loading.

An example of loading OKUMA LB10II lathe is shown in Fig. 11, following 'Set1' parameter values shown in Table 3.

6. CONCLUSIONS

Task based parametric programming of industrial robots is conducted off-line and results in substantial time savings. It is feasible when essentially the same task is assigned to a robot, a stereotypical situation referring to loading and unloading a machine tool. In this case, differences pertain to geometry of the part that is handled and the machine tool that is served, so these are the parameters at hand. In addition, an end effector tool universally employed in this kind of task is needed. The task is broken down into subtasks and each of them is assigned a section of the path. Path points are topologically the same, only differing in terms of coordinates. These are ideally computed analytically, but if this is cumbersome or too complex, CAD-based computation can be substituted. Kinematic simulation on any CAD system supporting constraints is normally necessary in order to check the path and safeguard it against collisions. However, the exact position and orientation of the robot with respect to the machine being tended needs to be registered, ideally by photogrammetry or equivalent high resolution technique.

The approach can be extended to cover other types of machine tools and other robots, too.

REFERENCES

[1] X. V Gogouvitis, G.-C. Vosniakos, *Construction of a virtual reality environment for robotic manufacturing cells*, Int. J. Comput. Appl. Technol., vol. 51, no. 3, 2015, pp. 173–184.

[2] S. Michas, E. Matsas, G.-C. Vosniakos, *Interactive programming of industrial robots for edge tracing using a virtual reality gaming environment*, Int. J. Mechatronics Manuf. Syst., vol. 10, no. 3, 2017, pp. 237–259.

[3] E. Rutten, L. Marc, *An imperative language for task-level planning : definition in temporal logic*, Artif. Intell. Eng., vol. 8, 1994, pp. 235–251.

[4] P. Rogalinski, *An approach to automatic robots programming in the flexible manufacturing cell*, Robotica, vol. 12, 1994, pp. 263–279.

[5] M. Prinz, H. C. Liu, B. O. Nnaji, T. Lueth, *From CAD-based kinematic modeling to automated robot programming*, Robot. Comput. Integr. Manuf., vol. 12, no. 1, 1996, pp. 99–109.

[6] S. Benbernou, *Factorization model of robotic tasks*, Artif. Intell. Eng., vol. 13, no. 1, 1999, pp. 11–20.

[7] N. Kontolatis, G. Vosniakos, K. Kyriakopoulos, *On parametric toolpath design of a robot serving a press-brake*, in 19th Flexible Automation and Intelligent Manufacturing, pp. 137–146, Middlesborough, UK, F. Nabhani (ed.), July 2009.

[8] G.-C. Vosniakos, P. Sierros, *Flexible design, analysis and programming methodology of robotic welding cells for multiple configuration products*, Int. J. Mater. Struct. Integr., vol. 1, no. 1, 2007, pp. 212–237.

[9] G.-C. Vosniakos and A. Chronopoulos, "Industrial robot path planning in a constraint-based computer-aided design and kinematic analysis environment," *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 223, no. 5, pp. 523–534, 2009.

[10] Z. Pan, J. Polden, N. Larkin, S. Van Duin, J. Norrish, *Recent progress on programming methods for industrial robots*, Robot. Comput. Integr. Manuf., vol. 28, no. 2, 2012, pp. 87–94.

[11] R. Datta, K. Deb, *Multi-objective design and analysis of robot gripper configurations using an evolutionary-classical approach,* 13th annual conference on Genetic and evolutionary computation, pp. 1843-1850, Dublin, Ireland, N. Krasnogor (ed.), July 2011, ACM.

[12] Adept Techn. Inc, *V + Language Reference Guide*. 1997.