Proceedings in Manufacturing Systems, Volume 17, Issue 3, 2021, 83-88 ISSN 2067-9238

USE OF A MODIFIED VIRTUAL MODEL OF A ROBOT ARM FOR THE CORRECTION OF POSITIONAL ERRORS CAUSED BY THERMAL DEFORMATIONS

Cozmin CRISTOIU[1,*], **Mario IVAN**[2], **Laurentiu STAN**[3]

[1)] Lecturer, PhD., Eng., Robots and Manufacturing Systems Department, University "Politehnica" of Bucharest, Romania
Postdoctoral student at Faculty of Industrial Engineering and Robotics, University "Politehnica" of Bucharest
[2)] Lecturer, PhD., Eng., Robots and Manufacturing Systems Department, University "Politehnica" of Bucharest, Romania
[3)] Student, Robots and Manufacturing Systems Department, University "Politehnica" of Bucharest, Romania

Abstract: *For some models of industrial robot arms, due to the motors located inside or internal mechanisms that can generate heat, after a working period from the start of the robot, structural elements can heat up causing slight deformations of the robot. Because of the serial structure of the kinematic chain, expansions and torsions of structural elements lead to errors that cumulate towards the endpoint of the robot. This paper proposes the usage of a modified virtual model of a robot that is modeled closed to the actual deformed model to compute the right angular values for the joints of the deformed robot in order to still reach the initial programmed targets.*

Key words: *industrial robot, heating, errors, compensation.*

1. INTRODUCTION

This stage is like a "puzzle piece" that must be placed in one of the empty spots of the global picture of a complete methodology towards improving the precision of an industrial robot arm by reducing thermally induced errors. On this research trip, two separate paths were taken:

A. Theoretical and experimental research to determine the thermal behavior of a robot arm.
B. Theoretical and experimental research to find out a software compensation solution for thermally induced errors.

During operation, the robot is going through two stages in which the errors vary differently.

b1. The first stage is the warm-up period of the robot where the structure is continuously deforming thus continuously affecting the induced errors.

b2. The second stage is that of thermal stabilization, in which the deformations of the robot stop, and the errors remain constant.

Because all geometric models of robot arms involve constant geometric parameters, it is more natural to think of a thermal compensation solution first for the second stage of thermal stabilization. Unfortunately, geometric models implemented in robot controllers are not editable so a software workaround must be found to compute joint values for a modified geometric model leading to the following sub step b2.1:

b2.1. design a thermally deformed virtual model of the robot arm and find a method so it can be used to compute joint angles by applying Inverse Kinematics to the deformed model and not on the ideal geometric one. Regardless of the method used to design and compute the Inverse Kinematics (IK) on the deformed virtual model it must be realized considering real temperature values and displacements meaning that the following two steps must be performed first:

b2.2. Measurements of robot temperature after warm-up.

b2.3. Quantitative evaluation/determination of the positioning error after warm-up.

For the robot model ABB IRB 140 (which is in the faculty laboratory), steps b2.2 and b2.3 were already studied previously in [3, 4, 5]. Having said that, the work therefore refers to the realization of the virtual model of the deformed robot and the application of IK to identify the necessary angles so that the programmed points can be reached even after the deformation of the robot. For this, CoppeliaSim (former V-Rep) robotic simulator will be used.

2. DEFINING THE VIRTUAL MODEL

The robotics simulator CoppeliaSim, with integrated development environment, is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution. The program has a library of models, among which you can also find ABB IRB 140. This virtual model is, however, taken from the robot manufacturer and includes dimensions of the elements and positioning of the couplings exactly as in reality and as they are defined on the ideal geometric model that use the robot controller (Fig. 1).

Unfortunately, this positioning of the joints does not correspond to the real model, although from the point of view of mathematical modeling and kinematics it is

---

[*] Corresponding author: Splaiul Independenței 313, district 6, 060042, Bucharest, Romania,
Tel.: 0040 21 402 9174,
E-mail addresses: *cozmin.cristoiu@gmail.com* (Cristoiu C.)

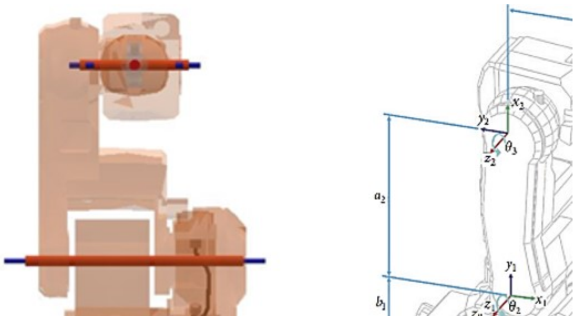

Fig. 1. Joint placement in original model in CoppeliaSim (and most of geometric models of the robot used in literature) [6].

irrelevant up to a point. The case in which the positions of the joints are relevant is the one in which the robot suffers deformations and then the geometric model should allow the addition of some parameters to represent these deformations. In CoppeliaSIM it is also possible to create a virtual model of a robot using Computer Aided Design (CAD) files for the visual representation of the elements and most importantly, the possibility of defining the couplings in any location. So, even if the structure of the robot looks approximately the same as the initial one (the geometric elements are used only for the graphic representation) in the case of the model presented in Fig. 2, the robot's joints were placed as in reality (considering the asymmetric structure of the robot) and taking into account and the displacements from their initial position considering the deformations of the robot structure previously determined in previous works [3, 4, 5]. Linear and angular displacements applied to each joint are presented in Tables 1 and 2.

These values were added to the initial positions of each joint conducting to the construction of the deformed virtual model of the robot. In Fig. 2, the warping of the robot caused by introduction of the deformation parameters can be observed in the front-view of the robot where the deformations were magnified 20 times to be observed with the naked eye.

Table 1

Joints linear displacement values

	Dx [mm]	*Dy* [mm]	*Dz* [mm]
J1	-0.0200698	0.000154	-0.05297
J2	0	0	0
J3	0.1008636	0.1148552	-0.001420766
J4	-0.0044	0.0241	-0.0149
J5	0	0	0
J6	-0.0162926	0	0

Table 2

Joints angular displacement values

	Dox [º]	*Doy* [º]	*Doz* [º]
J1	0	-0.011	-0.0137
J2	0.0108	0	0
J3	0	0.0081	0.0098
J4	0	0	0
J5	0	0	0
J6	0	0	0

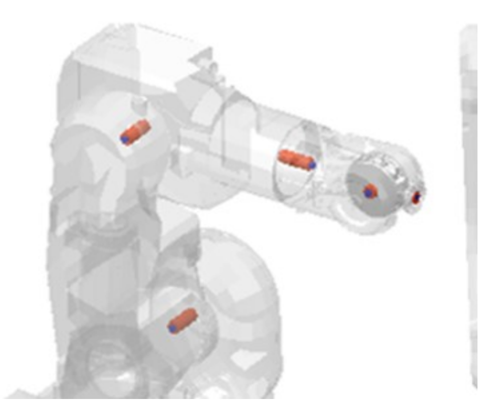

Fig. 2. Custom virtual model that considers real position of robot joints and displacements of their initial position caused by thermal deformation of the structure.

This model was further used for IK computation as if it were the real robot at the time of thermal stability which intuitively can be said to no longer be able to hit the targets as precisely if the coupling angles are not slightly adjusted or recalculated.

3. SIMULATION PROCEDURE

By default, CoppeliaSim can compute IK for the initial virtual model using pseudo-inverse IK method [7] determining each joint angle for the robot to reach the programmed targets. In order to avoid manually defining of each robot target (usually robot programs can consist in a large number of points) and recording all joints values for every robot pose at each programmed target, a script was developed in LUA language in order to automate previously mentioned tasks. The script needs to be fed with a .csv file containing the coordinates of the, the duration of the simulation (otherwise when all points are covered the simulation stops) and a specified precision. The simple logic block is depicted in Fig. 3.

Robot targets were imported from the initial experimental tests regarding robot status check, calibration, and thermal behavior recording. The points

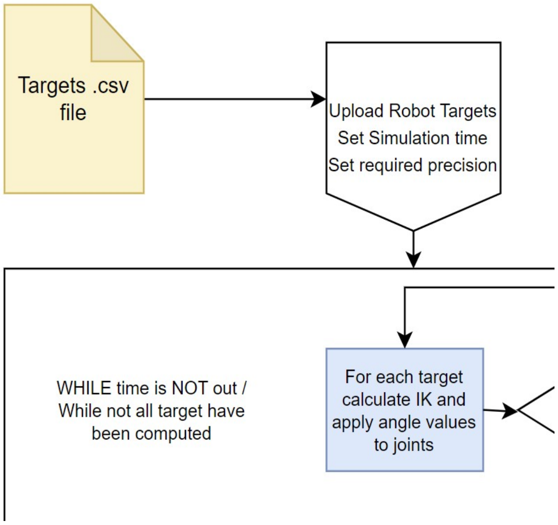

Fig. 3. Logic diagram of the script.

were programmed considering the space available for the robot (given that it is closed in an enclosure within the laboratory) and taking into account the recommendations of ISO 9283. A total number of 72 equidistant points were defined, 45 of which correspond to the area in front of the robot and 27 points to the right of the robot.

Coordinates of each target are presented in Tables 3 and 4.

These targets were used to determine the status of the robot and to calibrate it. A target is not only defined by Cartesian coordinates but also by its orientation. Usually, these articulated arm robots can touch points in various configurations and with different orientations of the characteristic point. The IK calculation methods usually provide multiple solutions from which the desired ones can be chosen (with the exception of cases where, due to singularities, no solutions are found). Therefore, the software compensation solution must allow the deformed robot to reach the programmed points without altering the configuration/orientation with which these points are reached. In order to reach the programmed points, 6 angles are generated through IK, one for each coupling of the robot, for each of the solutions and configurations found. It is obvious that for the deformed model, the identified angles will be slightly different (this is actually what we want to find). So we know the coordinates of the points and the orientation with which they must be reached. What we do not know at the moment are the angles calculated for the ideal (undeformed) model for certain particular configurations. For this reason, the points were defined as targets in RoboDK (online/offline simulation and robot programming application) which has integrated both the postprocessor for the ABB IRB 140 robot controller and the ability to extract joint angles, the position of the TCP (Tool Center Point) and its orientation at every target. Most of the needed information can be read out at any moment from the RDK robot control pannel (Fig. 4).

Each target was intentionally defined with the same orientation (every corresponding reference frame is the same) having only different coordinates. Two

Table 3

Front cube targets

Point	*X*	*Y*	*Z*	Point	*X*	*Y*	*Z*
P1	400	300	250	**P23**	550	0	400
P2	400	150	250	**P24**	550	-150	400
P3	400	0	250	**P25**	550	-300	400
P4	400	-150	250	**P26**	550	-300	250
P5	400	-300	250	**P27**	550	-150	250
P6	400	-300	400	**P28**	550	0	250
P7	400	-150	400	**P29**	550	150	250
P8	400	0	400	**P30**	550	300	250
P9	400	150	400	**P31**	700	300	250
P10	400	300	400	**P32**	700	150	250
P11	400	300	550	**P33**	700	0	250
P12	400	150	550	**P34**	700	-150	250
P13	400	0	550	**P35**	700	-300	250
P14	400	-150	550	**P36**	700	-300	400
P15	400	-300	550	**P37**	700	-150	400
P16	550	-300	550	**P38**	700	0	400
P17	550	-150	550	**P39**	700	150	400
P18	550	0	550	**P40**	700	300	400
P19	550	150	550	**P41**	700	300	550
P20	550	300	550	**P42**	700	150	550
P21	550	300	400	**P43**	700	0	550
P22	550	150	400	**P44**	700	-150	550
P23	550	0	400	**P45**	700	-300	550

Table 4

Side cube targets

Point	*X*	*Y*	*Z*	*Point.*	*X*	*Y*	*Z*
P1	-150	-400	300	**P15**	150	-550	450
P2	0	-400	300	**P16**	150	-550	300
P3	150	-400	300	**P17**	0	-550	300
P4	150	-400	450	**P18**	-150	-550	300
P5	0	-400	450	**P19**	-150	-700	300
P6	-150	-400	450	**P20**	0	-700	300
P7	-150	-400	600	**P21**	150	-700	300
P8	0	-400	600	**P22**	150	-700	450
P9	150	-400	600	**P23**	0	-700	450
P10	150	-550	600	**P24**	-150	-700	450
P11	0	-550	600	**P25**	-150	-700	600
P12	-150	-550	600	**P26**	0	-700	600
P13	-150	-550	450	**P27**	150	-700	600
P14	0	-550	450				

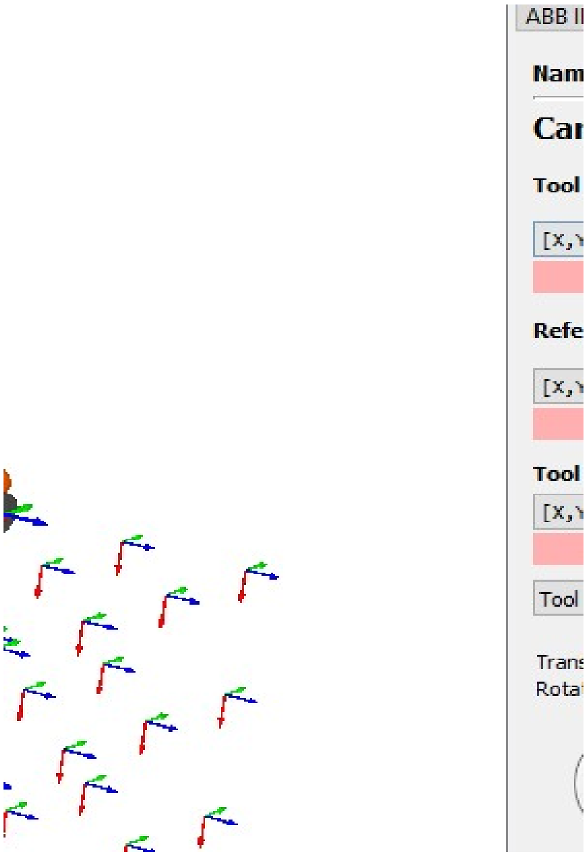

Fig. 4. RoboDK robot control panel.

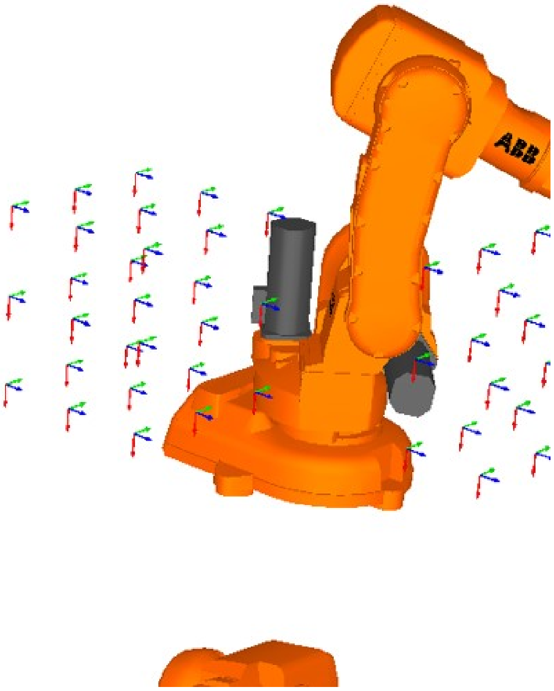

Fig. 5. Programmed targets in RoboDK

configurations were selected to reach the targets, one for the front cube and one for the side cube. Representation of each target defined in RoboDK and both of the robot configurations are presented in Fig. 5.

All coordinates, joint angles, and TCP-orientation at each target were recorded in .CSV tables. Due to the large format of the table, TCP data is presented only for the first 5 targets from each cube). First 5 points from the front cube are presented in Table 5.

First 5 points from the side cube are presented in Table 6.

These values are the real ones that are sent to the robot controller when the program is executed and are taken as control data.

Now to be able to check the script and the model made in CoppeliaSim, the simulation in must be executed twice.

Table 5

Example of TCP data at front cube targets

Target Coordinates [mm] - RDK			TCP orientation [°] - RDK			Joint angles [°] - RDK		
X	**Y**	**Z**	**Ox**	**Oy**	**Oz**	**1**	**2**	**3**
400	300	250	0	90	0	41.85	44.63	25.88
400	150	250	0	90	0	24.12	40.71	39.86
400	0	250	0	90	0	0	39.84	44.97
400	-150	250	0	90	0	-24.12	40	39.86
400	-300	250	0	90	0	-41.85	44.63	25.88
X	**Y**	**Z**	**Ox**	**Oy**	**Oz**	**4**	**5**	**6**
400	300	250	0	90	0	43.53	-75.61	-13.29
400	150	250	0	90	0	24.41	-81.4	-3.88
400	0	250	0	90	0	0	-84.81	0
400	-150	250	0	90	0	-24.41	-81.4	3.88
400	-300	250	0	90	0	-43.53	-75.61	13.29

Table 6

Example of TCP data at side cube targets

Target Coordinates [mm] - RDK			TCP orientation [°] - RDK			Joint angles [°] - RDK		
X	***Y***	***Z***	***Ox***	***Oy***	***Oz***	***1***	***2***	***3***
150	-700	600	0	90	0	-83.07	45.14	-44.24
0	-700	600	0	90	0	-95.30	44.67	-43.46
-150	-700	600	0	90	0	-107.07	51.86	-55.73
-150	-700	450	0	90	0	-107.07	55.62	-39.54
0	-700	450	0	90	0	-95.30	50.24	-29.88
X	***Y***	***Z***	***Ox***	***Oy***	***Oz***	***4***	***5***	***6***
150	-700	600	0	90	0	90.10	83.07	269.09
0	-700	600	0	90	0	89.88	95.30	268.78
-150	-700	600	0	90	0	91.18	107.03	274.05
-150	-700	450	0	90	0	85.13	106.38	253.22
0	-700	450	0	90	0	88.14	94.97	249.55

Once, without applying deformations to the virtual model, to check if and how close the solutions identified in CoppeliaSim are to the initial (control) ones generated by RoboDK. The second time, the simulation in CoppeliaSIM must be carried out using the deformed model (applying the deformations of the virtual model that will be read from a .CSV table) to check if IK is still generating solutions so that the robot can reach the points (and with what precision) keeping the TCP configuration and orientation. The results of both simulations are also recorded in. CSV tables in the same way as presented earlier so that the differences can be easily calculated.

4. RESULTS

During the simulation, the inverse kinematics is calculated for each successive point and the virtual model is moved to the corresponding position. For the side cube targets the same configuration (as in RoboDK) of the robot can be observed in Fig. 6.

Being an iterative method, the IK stops when a solution is found within a specified (desired) precision or when a specified timeout counter is surpassed. Values of these parameters are to be empirically adjusted until

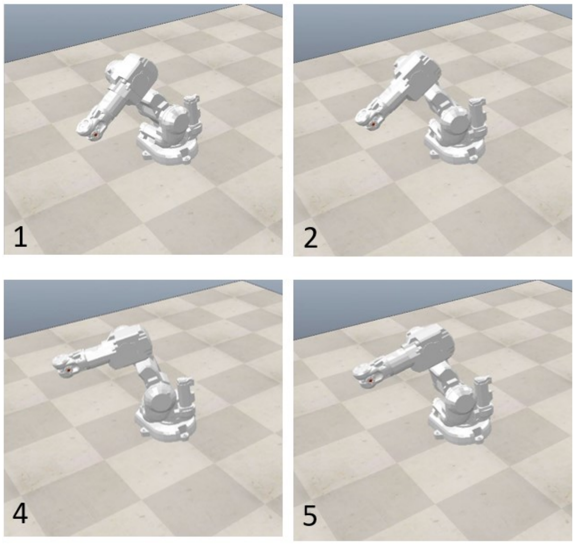

Fig. 6. Configuration found for side cube targets.

satisfactory solutions are found. Every pose of the robot is recorded in an output results file with the same structure as the input one. Due to the large dimensions of tables, for the undeformed case simulation, only maximum deviations are presented in Tables 7 and 8.

Results for front cube targets using the deformed model are presented in Table 9.

Table 7

Maximum deviations, front cube – undeformed model

	Coordinate deviations			Orientation deviations		
MAX	0.004643	0.004223	0.009644	0	0.02	0.001
MIN	4.77E-05	-0.00304	-0.00099	0	0	-0.001

Table 8

Maximum deviations, side cube – undeformed model

	Coordinate deviations			Orientation deviations		
MAX	400	150	600	0	90	0.001
MIN	-150	-700	-0.00099	0	0	-0.001

Table 9

Results for front cube targets – deformed model

	Coordinates			*TCP orientation*		
	X	**Y**	**Z**	**Ox**	**Oy**	**Oz**
1	400	299.9999	250.0001	0	90	-0.001
2	400.0001	149.9999	250	0	90	-0.001
3	400	3.35E-05	249.9999	0	90	-0.001
4	400.0001	-150.0001	250	0	90	0
5	399.9999	-300	249.9999	0	90	0
6	400.0001	-299.9999	400.0001	0	90	-0.001
7	399.9999	-150	400	0	90	0
8	400.0001	0.000067	399.9999	0	90	0
9	400	150	399.9999	0	90	-0.001
10	400	300	400.0001	0	90	0
11	400	300	550.0001	0	90	0
12	400	150.0002	550.0002	0	90	0
13	400	0.000067	550	0	90	0
14	399.9999	-149.9998	550.0001	0	90	0
15	400.0001	-300	550.0001	0	90	-0.001
16	550.0001	-300	549.9998	0	90	-0.001
17	550	-149.9999	550.0001	0	90	-0.001
18	550	9.68E-05	550.0001	0	90	0
19	550	149.9999	550.0002	0	90	0
20	550.0001	300	550.0002	0	90	-0.001
21	549.9999	300.0001	400	0	90	-0.001
22	550	150.0001	400.0001	0	90	0
23	550.0001	7.4E-06	399.9999	0	90	-0.001
24	549.9996	-150.0002	400.0001	0	90	-0.001
25	550	-300	400	0	90	0
26	549.9999	-300	250.0001	0	90	-0.001
27	550.0001	-150	250.0001	0	90	0
28	550	-5.97E-05	250	0	90	-0.001
29	549.9997	150	249.9997	0	90	-0.001
30	550.0003	300.0001	250	0	90	-0.001
31	700	300	250.0001	0	90	0
32	699.9999	150.0001	250	0	90	0
33	699.9997	0.000149	249.9999	0	90	-0.001
34	699.9997	-150	250	0	90	-0.001
35	699.9944	-299.9994	249.9916	0	90	-0.002
36	699.9999	-300	400	0	89.98	0
37	700	-150	399.9999	0	90	-0.001
38	699.9997	-0.000142	399.9999	0	90	-0.001
39	699.9997	150.0002	399.9999	0	90	-0.001
40	699.9999	300	400.0001	0	90	0
41	699.9999	300	550	0	90	-0.001
42	700.0001	150.0002	549.9998	0	90	-0.001
43	699.9999	-7.5E-06	549.9999	0	90	0
44	700	-149.9999	549.9998	0	90	0
45	700	-300	550	0	90	0

Table 10

Results for side cube targets – deformed model

	Coordinates			**TCP orientation**		
	X	**Y**	**Z**	**Ox**	**Oy**	**Oz**
1	150	-699.9999	600.0003	0	90	-0.001
2	-1.5E-05	-699.9997	600	0	90	-0.001
3	-150	-699.9999	600	0	90	-0.001
4	-150	-700	449.9998	0	90	0
5	2.23E-05	-699.9998	450.0001	0	90	-0.001
6	150	-699.9999	450	0	90	-0.001
7	150.0001	-700.0002	299.9998	0	90	0
8	3.72E-05	-700	300.0002	0	90	-0.001
9	-150	-699.9999	300.0001	0	90	-0.001
10	-150	-550.0001	300	0	90	0
11	0.000067	-550.0001	300	0	90	0
12	149.9999	-549.9999	299.9999	0	90	-0.001
13	150	-549.9998	449.9999	0	89.98	-0.001
14	-0.00025	-549.9998	450.0001	0	90	0
15	-150	-549.9997	449.9999	0	90	0
16	-150	-550.0001	600.0001	0	90	0
17	2.98E-05	-550.0001	600.0001	0	90	-0.001
18	150	-550.0001	600.0001	0	90	-0.001
19	149.9999	-400.0001	600.0001	0	90	-0.001
20	-6E-05	-400.0001	600	0	90	0
21	-150	-400.0001	600	0	90	0
22	-150	-399.9999	449.9999	0	90	0
23	1.49E-05	-399.9999	450.0001	0	90	-0.001
24	150	-399.9999	449.9999	0	90	0
25	150	-400	300.0001	0	90	-0.001
26	0.000119	-399.9996	300	0	90	-0.001
27	-150	-400	299.9998	0	90	-0.001

Table 11

Maximum deviations, front cube – deformed model

	Coordinate deviations			Orientation deviations		
MAX	0.005615	0.000229	0.008434	0	0.02	0.002
MIN	-0.00025	-0.00058	-0.00019	0	0	0

Table 12

Maximum deviations, side cube – deformed model

	Coordinate deviations			*Orientation deviations*		
MAX	0.000253	0.000167	0.000197	0	0.02	0.001
MIN	-0.00013	-0.00044	-0.00032	0	0	0

Results for front cube targets using the deformed model are presented in Table 10.

Maximum coordinates and orientation of the deformed model for front cube targets are presented in Table 11.

Maximum coordinates and orientation of the deformed model for side cube targets are presented in Table 12.

5. CONCLUSIONS

Geometric models from industrial robot controllers cannot be edited by typical users (but only by their development teams) and they rely on mathematical models that use constant parameters for lengths of structural elements and initial positioning of their joints. These geometric models are depicting the theoretically ideal robot (non-deformable and without being subject to errors) but in the reality every robot (as any electro-mechanical assembly) is affected by errors among which errors caused by thermal deformations. For some robotic applications (such as assembly of electronic components or robotic machining) the positioning performance of a robot is very important as it is reflected in the quality of the products obtained so increasing the robot precision is very desirable. The solution proposed in this paper is aiming to eliminate or reduce as much as possible the errors that appear due to the heating and deformation of the robot. The idea is to use a separate software application and algorithm (rather than using only the robot control system) in order to 3D model and compute the IK of a virtual deformed model without using usual Denavit-Hartenberg (DH) [8] convention. The IK must be computed on a custom geometric model that includes 6 supplementary parameters for each joint representing each linear and angular deviation caused by the thermal deformation (along/around XYZ axis) and detected by experimental measurements. The use of the CoppeliaSim application and of the created script essentially represents the post-processing of the targets defined in a robot program in order to obtain modified values of joint angles so that the deformed robot to still reach the targets. In essence, the most important aspect is that the angles calculated for the robot joints to lead to the positioning of the TCP without large deviations from the initial programmed coordinates and without orientation deviations. The maximum deviations obtained after the IK calculation using CoppeliaSim and the mentioned script are less than 10 microns and 0.02 degrees. Considering that after the warm-up of the robot, the TCP deviation determined in [5] were about 97 microns, a decrease of up to 10 microns represents a reduction of over 89% of the error caused by thermal deformation. With the mention that there are signs of possible improvement (by increasing the number of iterations or the waiting time limit for calculating the solutions through IK) the presented method represents a viable offline software solution for thermal error compensation (only after the thermal stabilization of the robot). From here on, the efforts will be primarily oriented towards finding a compensation solution and for the transient thermal period of the robot.

Acknowledgments: These studies were carried out as a research activity within the postdoctoral research project of the first author. This work was supported by the grant POCU/993/6/13/153178, "Performanță în cercetare" (Research performance) co-financed by the European Social Fund within the Sectorial Operational Program Human Capital 2014–2020".

REFERENCES

[1] C. Mavroidis, S. Dubowsky, P. Drouet, J. Hintersteiner, J. Flanz, *A systematic error analysis of robotic manipulators: application to a high-performance medical robot*, Proceedings of International Conference on Robotics and Automation, IEEE, Vol. 2, 1997, pp. 980–985.

[2] C. Mehdi, K. Jean-Yves, B. Alex, *Thermal aspects on robot machining accuracy*, Proceedings of IDMME – Virtual Concept 2010, France 2010.

[3] A.F. Nicolescu, C. Cristoiu, C. Dumitrascu, R. Parpala, *Recording procedure of thermal field distribution and temperature evolution on ABB IRB 140 industrial robot*, IOP Conference Series: Materials Science and Engineering, IOP Publishing, Vol. 444, No. 5, 2018, p. 052023.

[4] C. Cristoiu, M. Zapciu, A.F. Nicolescu, C. Pupaza, *Thermal deformation analysis of ABB IRB 140 Industrial Robot,* UPB Scientific Bulletin, Series D: Mechanical Engineering, Vol. 82, Issue 2, 2020, pp. 61–72.

[5] C. Cristoiu, *Research on the influence of the thermal behaviour of industrial robots on their performance*, PhD. Thesis, University "Politehnica" of Bucharest, 2020.

[6] David Alejandro Elvira-Ortiz et all. *Vibration Suppression for Improving the Estimation of Kinematic Parameters on Industrial Robots*, Hindawi Publishing Corporation Shock and Vibration, Vol. 2016.

[7] SR Buss, *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods*, Department of Mathematics, University of California, San Diego, 2009.

[8] A. Chennakesava Reddy, *Difference between Denavit - Hartenberg (d-h) classical and modified conventions for forward kinematics of robots with case study*, International Conference on Advanced Materials and manufacturing Technologies (AMMT), December 18-20, 2014.