Interface development dedicated to connecting CAD tools for 3D modeling of complex objects and industrial robot's controller

Uroš Ilić, Jovan Šumarac, Ilija Stevanović, Aleksandar Rodić

***Abstract*—This article covers the development of the interface dedicated to converting CAD models into some universal computer readable form – Excel spreadsheet. Main intention is transcription of complex models' geometry and coloring to numerical database. First the article will describe the process of saving CAD models in compatible form and its conversion to Microsoft Excel spreadsheets via a set of MATLAB's designated scripts and functions. Main focus is forming a set of significant points meant to guide the industrial robot's end effector to outline (with a laser pointer) the surface of the object that's in contact with the work desk. At the end, the given trajectory will be shown in MATLAB's plotter and correspondent Excel table.**

***Index Terms*—CAD; MATLAB; Microsoft Excel; Path planning; Industrial robots**

I. Introduction

MASS production as a leading example of XX century rapid development is slowly but consistently being replaced by mass customization. Its elementary feature is the use of flexible computer-aided manufacturing systems to produce custom output. Such systems combine the low unit costs of mass production processes with the flexibility of individual customization.

Even though integrating robots to manufacturing and assembly processes started long time ago, new problem opposes: How to program the robot in such way so production lines become more flexible in terms of fulfilling customer demands? For example, if it happens that there is a specific requirement on only one product, it is necessary to program the entire workstation in accordance with the requirement. After performing the task, it needs to be reprogrammed for the prior purpose. All this work is forming expenses, both the financial ones and also in form of human resources, i.e., the engagement of engineers and/or programmers.

With the advent of factory digitalization and Industry 4.0, the whole world is moving towards factories with no or minimal number of workers. For this purpose, we strive for remote control of production systems, i.e., robots, both on production tasks and on assembly lines. With that in mind, it is necessary to develop such system where it is possible to control individual robots (manipulators). One of the conceptual solutions is forming a database within the cloud, which could be accessed simultaneously by the robot, which performs the assembly, and a programmer, who would work on updating the database itself.

II. Objectives Of CAD Systems And Robotic Controller Integration

Each of the designed parts will, firstly, be modeled in Solidworks® environment, and then exported with the appropriate extension. Such files are imported into MATLAB and, afterwards, exported to Microsoft Excel spreadsheets. This type of data storing is found convenient for later use by numerous programming languages such as C, C++, Python or MATLAB itself once again. For each of the imported models in MATLAB, a set of designated points in space with TCP's approach vector will be formed.

The final goal is to generate the trajectory of TCP with the given information, and thus solve the inverse kinematic problem of the manipulator. In other words, it is necessary to form a sequence of matrices of significant points' coordinates along with approach vector's x-y-z components, which will unambiguously determine the position of the robot's segments when moving, i.e., performing a certain technological operation.

All gathered information will be integrated into the database and the robot will, based on the data it receives from the camera, with the help of artificial intelligence, define which parts are in front of it, and in which position and orientation. Based on all of that, the robotic system will choose the appropriate technological operation of assembly or manufacturing process. Finally, with the given information, the robot can initialize, first the formation of the trajectory, and afterwards its realization.

Uroš Ilić is with the Faculty of Mechanical Engineering, University of Belgrade, 16 Kraljice Marije, 11020 Belgrade, Serbia (e-mail: ilicuros01@gmail.com)

Jovan Šumarac is with the Institute Mihajlo Pupin, University of Belgrade, 15 Volgina, 11020 Belgrade, Serbia (e-mail: jovan.sumarac@pupin.rs)

Ilija Stevanović is with the Institute Mihajlo Pupin, University of Belgrade, 15 Volgina, 11020 Belgrade, Serbia (e-mail: ilija.stevanovic@pupin.rs

Aleksandar Rodić is with the Institute Mihajlo Pupin, University of Belgrade, 15 Volgina, 11020 Belgrade, Serbia (e-mail: aleksandar.rodic@pupin.rs)

III. Using Solidworks® Software For Designing Complex Objects

Solidworks® is a solid modeling computer-aided design (CAD) program that's suitable for designing numerous complex objects. In this particular case various mechanical components are modeled and used for development of the interface. Some of these objects are shown on Figure 1.

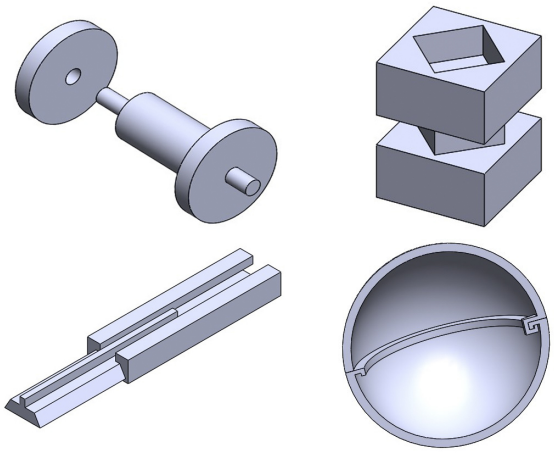

Fig. 1. Representation of typical complex objects found in the machine industry. These objects are chosen as they represent various kinds of assemblies, depending on interrelation of the parts.

The selected assemblies are some of the most common types in the machine industry and assembly processes. They are listed gradationally, so that simpler assemblies are described first. The simplicity of the assembly is observed from the aspect of the number of possible mutual positions that the parts can or must take to form an assembly.

A. Axle with bearings

This is one of the simplest mounting assemblies. In layman's terms, bearings can be found in infinitely many positions with the axle. This statement is justified by the axial symmetry of the parts that enter the assembly. It is necessary to achieve the appropriate positions immediately before the assembly process and perform a simple translation in order to form an assembly. Their model is shown in Fig. 1 in upper-left corner. Even though bearings need significant force to be mounted, this assembly is only used in the field of robot kinematics.

B. Simple puzzle

This form represents a more complex assembly than the previous one since only 4 mutual positions in the assembly are possible. They are achieved by a 90° rotation around the mounting axis. If the parts are not marked, the observer will not even be able to distinguish the different positions. Even though this type of the simplest puzzle is not typically found in the industry it's just a mere representation of assembly of prismatic shapes. It is shown in Fig. 1 in upper-right corner.

C. Linear guide rail

Classical example of a linear bearing, that is actually an assembly that can only be formed in a certain way. The guide and the slider must be brought to a precisely determined position and orientation in order for the assembly to be formed by a simple translation or inserting the slider into the guide. This assembly is shown on Fig. 1, bottom-left corner.

D. Spherical vessel

Unlike the other parts, it is not given in some of the standard views (isometry, trimetry ...). The given view (Fig. 1, bottom-right corner) is selected to show the grooves for closing and opening. This type of assembly is one of the most complex assemblies that can be found on the assembly line. It is necessary to bring both hemispheres to a position in which the tongue of the upper part coincides with the slit of the lower part. After connecting the hemispheres, it is necessary to rotate them by a certain angle and thus achieve the formation of the assembly, i.e., sealing the spherical vessel.

E. 3D puzzle pyramid

This type of assembly was taken into consideration as an example of the most complex technological assembly operation. The pyramid consists of 5 parts. It is necessary to assemble them in a specific order. An additional aggravating circumstance is the difficulty of distinguishing the parts of the pyramid itself. For this purpose, the outer sides of the parts are painted with various colors, while the inner sides are painted with the characteristic color of wood. The idea is to assemble the pyramid by a two-handed manipulator in some of the future works. Both assembled and disassembled pyramid is shown in Figure 2, respectively.

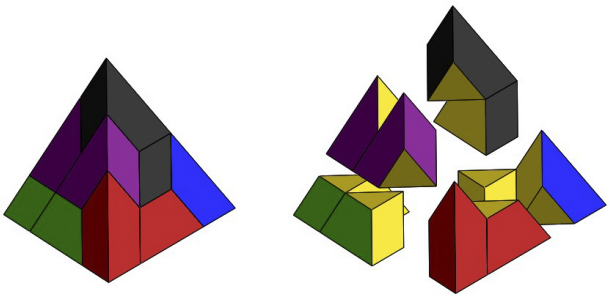

Fig. 2 – Comparison of assembled and disassembled pyramid. The complexity of parts' interrelationships can be seen on the figure.

IV. Surface Triangulation, Data Export And Objects' Database Creation

In order for the parts and its CAD models to be readable by the robot's control unit, it is necessary that the geometry data of those parts is stored in numerical format. For this purpose, Microsoft Excel spreadsheet is chosen. Apart from being compatible with numerous programming languages, it is also easy to be reviewed by the user of the interface.

Of course, direct conversion from CAD models to Excel spreadsheets is not possible. Therefore, the MATLAB

software package will be used, where the procedures for the indirect conversion of parts from the Solidworks® environment into an Excel spreadsheet will be written. In order for all this to be feasible, it is necessary to export the CAD model in the appropriate format. Figure 3 represents the flowchart of this method.

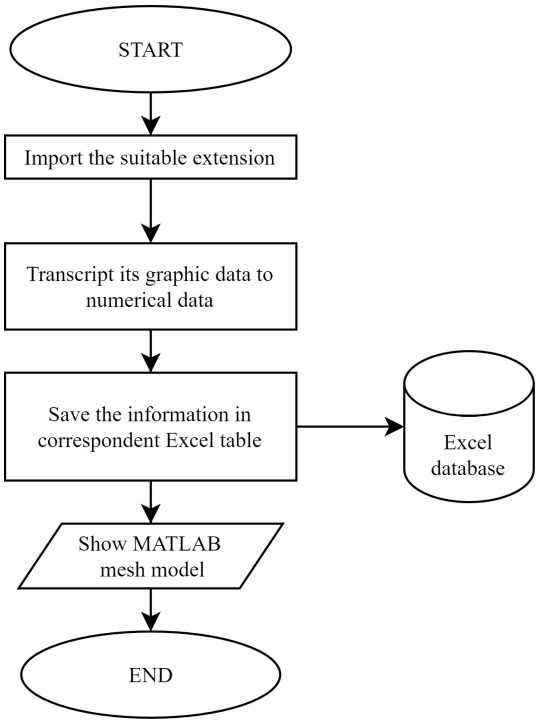

Fig. 3 – Flowchart representing the process of transferring the CAD model to the designated Excel spreadsheet with a mesh model's representation in MATLAB plotter

Of all the possible extensions that Solidworks® supports, only two solutions are significant - STL format (.stl) and VRML format (.wrl). Although the selection of each of them entails certain advantages and disadvantages, that are presented in Table 1.

TABLE I
COMPARISON OF CHOSEN EXTENSIONS' CHARACTERISTICS

Format (extension)	Advantages	Disadvantages
VRML (.wrl)	Ability to work with colors and their RGB vectors	Poorly researched topic; Lack of support on MATLAB forums
STL (.stl)	Relatively easy import to MATLAB; Very strong support within the MATLAB community	Able to store only geometry data, but not coloring of the faces

A. STL files

The abbreviation STL (Standard Triangle Language) represents a format suitable for 3D printing, i.e., stereolithographic methods in general. The format is characterized by triangulation of surfaces, i.e., division of surfaces into triangles. Figure 4 shows the process of forming a STL file.

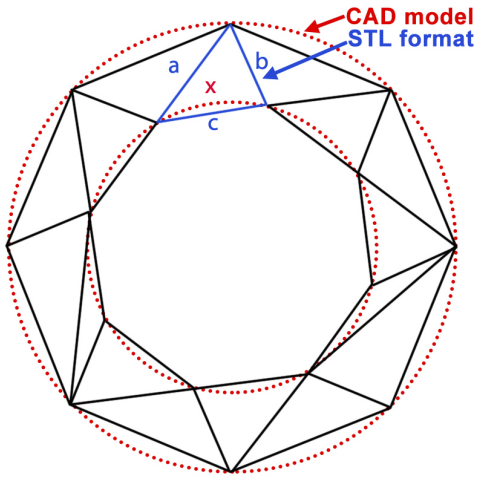

Fig. 4 – From the amorphous structure of the torus shape represented with dotted red line in the picture, a multitude of triangles is formed (one of which is marked in blue).

If it is amorphous, i.e., the more complex the surface, the greater the number of triangles in the network and vice versa. It is quite certain that simple geometric surfaces such as rectangles or squares will be divided into only two triangles, while circles and elliptical shapes will be divided into a finitely large number of triangles. Every STL file is characterized with connectivity matrix, coordinates of every point and each triangle's normal vector. The connectivity matrix has dimensions 3xn, where n represents number of triangles. This matrix defines what points form what triangle, respectively. Each triangle has its own corresponding normal vector represented with a unit vector.

B. VRML files

The VRML (Virtual Reality Modeling Language) format was created at the end of the last century with the intention of displaying three-dimensional vector graphics objects. The main emphasis was on the implementation of this format in internet communication, especially web presentations. These types of files (although their extension is *.wrl) are actually textually formatted and very easy to read by humans. Of course, by reading only a text file, one cannot create the body image that the file describes. Therefore, MATLAB will be used, which will read certain segments of the file, convert them into numerical values and form from them, first a mesh model, and then an Excel spreadsheet.

As mentioned in Table 1, VRML files have the ability to store information about faces' coloring. Information about

coloring is stored in similar way like those for normal vectors. For each generated triangle a three-dimensional vector is formed that describes primary colors in additive color synthesis. Each component represents the amounts of red, green and blue, respectively.

C. *Creating the database*

After successfully exporting the models in suitable form, it's needed to import them in the MATLAB's environment. After loading the file's full name, the designated function makes the spreadsheet with the same name as the STL/VRML file. Given spreadsheet is formatted in such manner (merging cells, naming the titles, etc.) that is appropriate for following code to fill in the information about triangulation (connectivity matrix), coordinates, normal vectors and, regarding the VRML file, the coloring too.

When reading of the chosen format is done, designated script stores significant values in appropriate matrices and starts filling in the Excel spreadsheet. If chosen format is STL, the script uses modified version of downloaded function stl_read(), made by MATLAB community. On the other side, if it's needed to convert the VRML file, the script calls designated function wrl_read(), that, firstly convert's the VRML file to text file and extracts desired strings of text that contains information that's needed to fill in the Excel spreadsheet. These strings are converted into numerical form and saved as matrices, that, as in previous case, fill in the database.

In Figure 5 a typical Excel spreadsheet with CAD model's info is represented. The spreadsheet is cut in half over the J-column in order to fit the formatting of this article.

	A	B	C	D	E	F	G	H	I	J
1	Triangles					Vertices (coordinates)				
2	Number	Vertices (number of vertex)				Number	X	Y	Z	
3	1	7	2	9		1	0	-84	0	
4	2	9	2	5		2	0	0	0	
5	3	4	6	9		3	8.083	-70	22.862	
6	4	5	2	4		4	8.083	-42	22.862	
7	5	4	2	1		5	8.083	-14	22.862	
8	6	4	1	3		6	24.249	-42	0	
9	7	2	7	1		7	24.249	-14	0	
10	8	1	7	6		8	32.332	-56	22.862	
11	9	1	6	10		9	32.332	-28	22.862	
12	10	9	6	7		10	48.497	-56	0	

	J	K	L	M	N	O	P	Q	R	S
1		Normal vectors (unit vectors)					Triangles' colors			
2		Number	X	Y	Z		Number	R	G	B
3		1	0.471405	0.816497	0.333333		1	1	0.784314	0
4		2	0.471405	0.816497	0.333333		2	1	0.784314	0
5		3	0.471405	-0.816497	0.333333		3	1	0.784314	0
6		4	-0.942809	0	0.333333		4	1	0.784314	0
7		5	-0.942809	0	0.333333		5	1	0.784314	0
8		6	-0.942809	0	0.333333		6	1	0.784314	0
9		7	0	0	-1		7	1	0.784314	0
10		8	0	0	-1		8	1	0.784314	0
11		9	0	0	-1		9	1	0.784314	0
12		10	0.942809	0	-0.333333		10	1	0.784314	0

Fig. 5 – The appearance of the Excel spreadsheet that originated from model's VRML file. This screenshot shows data only for first 10 triangles, even though the number of triangles can be measured in thousands for amorphous surfaces.

Here, a user can see from what vertices the triangles are format, e.g., the vertices numerated as 7,2 and 9 form the first triangle. Their coordinates in local coordinate system can be read in the neighboring table. Components of the normal unit vector for the first triangle can be read from cells L3 to N3 and its RGB coloring vector can be found in cells Q3:S3. Note that this spreadsheet came from conversion of the VRML file, since it has information about the coloring. The spreadsheet formed from STL files is practically the same without the last (fourth) table describing the colors.

Note that the tables always start from the same cell, no matter what model is being converted. Only change is number of rows in the Excel file. This feature enables making the universal MATLAB function for reading Excel spreadsheets and storing the values in corresponding matrices.

V. Generating Mesh Models From 3D Database

As a verification of the stored data, it's needed to read the spreadsheet and visualize that numerical information with a mesh model. If and only if the mesh model made by MATLAB plotter is same as initial CAD model, the code can be verified.

Firstly, the code calls the Excel file with desired name and starts reading the data from the tables. For every table read, it stores the values in corresponding matrix. After that, an already built-in functions form a three-dimensional mesh from the connectivity list (first table in the spreadsheet) and corresponding coordinates of given vertices (second table in the spreadsheet). Afterwards, each normal unit vector is drawn with the origin in triangle's centroid. If we're working with the STL files, all triangles will be painted with the same color. On the other side, if the user is converting the VRML file, each triangle will be colored with their correspondent RGB vector. Figure 6 shows the CAD model of an axle on the left and the result of reading the spreadsheet on the right. This is the result of converting the STL files.

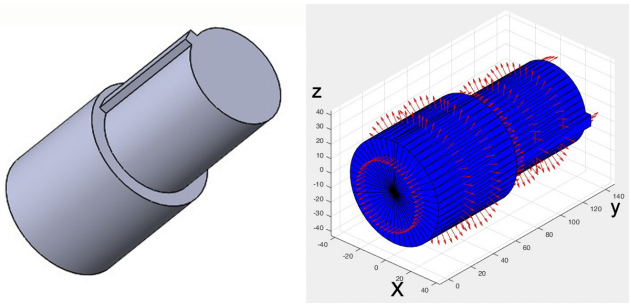

Fig. 6 – The result of the conversion of the CAD model to Excel spreadsheet and back again to MATLAB's mesh model. Coloring of the mesh model can be changed to any color, but whole model must be colored uniformly

On the other side, if we chose to export the CAD model as a VRML file, it will store its original coloring and represent it in its original colors in MATLAB plotter. Figure 7 shows a mesh model with surface vectors of 3D puzzle pyramid's red part in its original coloring. As it was said earlier the parts' outer surface is colored differently, but their inner sides depict wood's natural coloring.

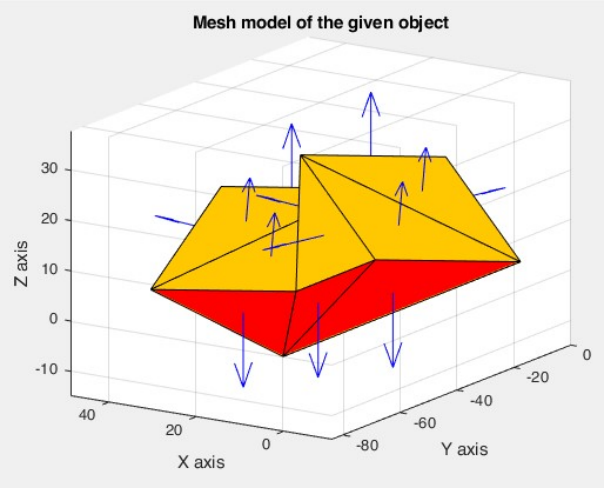

Fig. 7 – 3D puzzle pyramid's corresponding mesh model of the red part shown in MATLAB plotter. Outer face is colored with red and inner ones are with color of wood, just like we colored them in Solidworks®. Blue arrows pointing outwards represent normal unit vectors of each triangle.

VI. Defining The Contour Of The Object And Corresponding Significant Points

The procedure for defining the path that will enable the TCP to outline the surface lying on the desk is given on the figure 7 in form of the flow chart. This flowchart represents the second part of the whole process – the extraction of the numerical data of modeled parts.

The first step is to acquire the information of the object on the work desk. The camera will recognize the object lying on the desk and its position and orientation. Let's say that camera recognized the object on Figure 7. With the help of artificial intelligence, desired object's information will be found in the database. With the knowledge of position of the object in global, i.e., robot's coordinate system, object's local coordinate system will be transformed. Hence the vertices in contact with the desk will change. Therefore, we'll know the exact vertices lying on the desk, and their coordinates in global coordinate system.

The problem occurs if the shape of the face is oval or circle-shaped. Since the triangulation transforms the circle into a polygon, the center of the circle becomes the vertex that' included in all triangles. Even though it has its z-coordinate equal to zero, we must eliminate it too in order to form a closed polygon.

With the points extracted from the Excel spreadsheet, the 2D polygon of the surface is made. On Figure 8 the polygon is drawn with red lines with normal vectors pointing outside. The face represented on the Figure is actually the outer side of the puzzle's part – the red side.

When the polygon representing the contour of the object lying on the work desk is constructed, we advance to forming a trajectory of the tip of the end-effector of the robot. As said earlier, the laser pointer will be attached to the end-effector that will outline the contour of the object. In order to do such a thing, we must make another polygon with desired offset from the edges of the initial polygon.

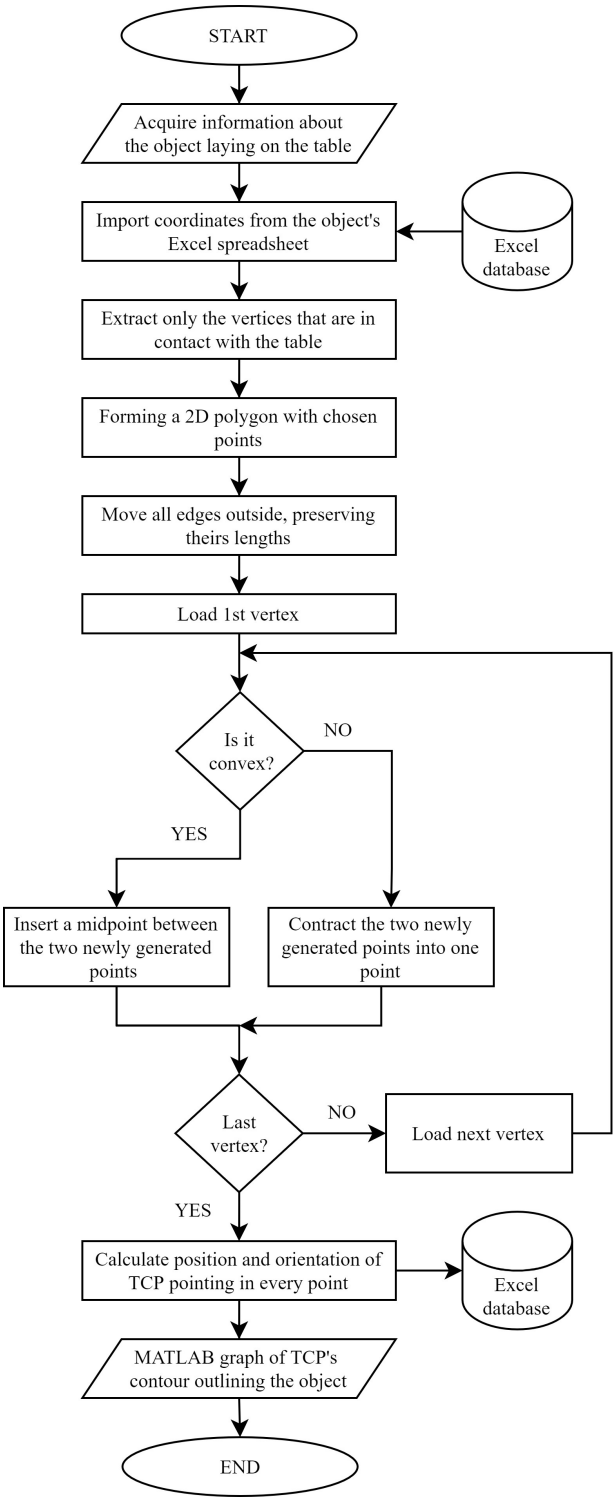

Fig. 7 – A flowchart depicting the process of the creation of significant points of TCP's path outlining the lying surface of the given object

The newly formed polygon is actually a modified enlargement of the surface's contour with rounded edges. It is constructed in a *for*-loop that goes over every point and checks if it's in a convex or concave vertex. The whole

process of creating the rounded enlarged polygon from the initial one can be described with the following steps:

- First, every edge is shifted outwards for desired distance. Now, instead of closed polygon, we have a set of disconnected lines, but with preserved lengths.
- For every vertex of the contour, algorithm checks whether they're convex or concave
- If they are convex, a new middle point is added, so that end effector outlines the convex vertex in a radius
- If, on the other hand, the vertex is concave, two significant points are replaced with one and therefore radius is made again in order to avoid collision.

After this part of the algorithm completes the whole contour and the last vertex is formed, the potential trajectory of the TCP is plotted in 2D diagram, as represented in Figure 8.

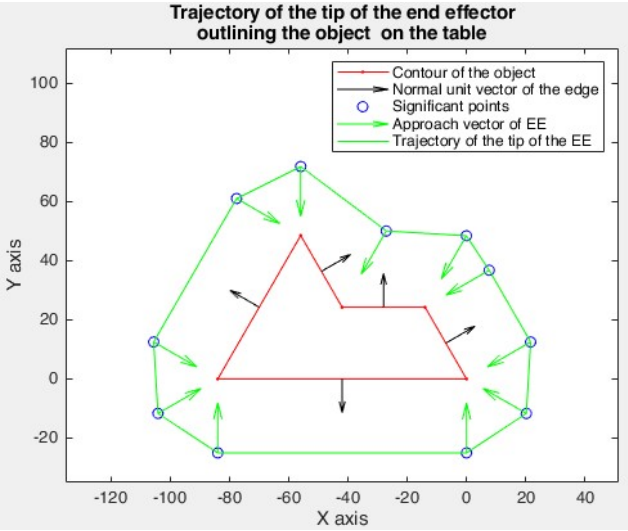

Fig. 8 – Newly generated trajectory of the end effector. It can be seen that one of the vertices is concave, hence a collapsing of neighboring points occurred. It can also be seen that the edges have been shifter outwards in the direction of normal unit vector.

In every, newly generated, significant point, the end effector's approach vector is computed. Approach vector will always have direction from the significant point to the correspondent point within the contour. Every point's position and approach vector's components are then stored in an Excel spreadsheet, as shown on Figure 9.

	A	B	C	D	E	F	G
1	Points	TCP Position			Approach vecotor		
2		X	Y	Z	X	Y	Z
3	1	0.00000	-25.00000	43.30127	0.00000	0.50000	0.86603
4	2	20.20032	-11.66273	43.30127	-0.43301	0.25000	0.86603
5	3	21.65070	12.49989	43.30127	-0.43301	-0.25000	0.86603
6	4	7.65070	36.74889	43.30127	-0.43301	-0.25000	0.86603
7	5	-0.04962	48.41162	43.30127	-0.25000	-0.43301	0.86603
8	6	-27.11194	50.03582	43.30127	-0.25000	-0.43301	0.86603
9	7	-56.00010	71.82244	43.30127	0.00000	-0.50000	0.86603
10	8	-77.65059	60.99708	43.30127	0.43301	-0.25000	0.86603
11	9	-105.65059	12.50008	43.30127	0.43301	-0.25000	0.86603
12	10	-104.20031	-11.66261	43.30127	0.43301	0.25000	0.86603
13	11	-84.00000	-25.00000	43.30127	0.00000	0.50000	0.86603

Fig. 9 – The look of the Excel spreadsheet containing the information about TCP's position (left part) and end-effector's approach vector outlining the object's surface on the table. The Z coordinates are the function of the desired angle of the laser's beam.

VII. Conclusion

Although the given example is a simple problem, it represents the new way of transcribing and eventually storing the numerical representation of complex geometry objects. This way of storing the data (in Excel spreadsheets) is chosen because it is the best way for a human eye to examine the data. This data can be stored in any file that is suitable for reading as long as the programmer knows how to store/look/extract the information.

It is clear that this process doesn't give homogeneous transformational matrices for every significant point. The idea is to store information in that way that can be easily retrieved by robot's controller or any other computing unit meant for simulation or robot control.

Acknowledgment

The results presented in the paper are obtained in the scope of the research projects: „Development and Experimental Performance Verification of Mobile Dual-Arms Robot for Collaborative Work with Humans", Science and Development Program – Joint Funding of R&D Projects of the Republic of Serbia and the People's Republic of China, contract no. 401-00-00589/2018-09, 2018-2021 and national R&D project no. TR-35003, both supported by the Ministry of education, science and technology development of Republic Serbia.

References

[1] Peddie, Jon, *The history of visual magic in computers: how beautiful images are made in CAD, 3D, VR and AR*, 1st ed, London, Springer,2013

[2] Tseng, M.M.; Jiao, J. (2001) "Mass Customization", in: *Handbook of Industrial Engineering, Technology and Operation Management* New York, NY: Wiley, 2001, 978-0-471-33057-8

[3] J. J. Uicker, G. R. Pennock, and J. E. Shigley, *Theory of Machines and Mechanisms*, New York, Oxford University Press, 2003

[4] Paul Festa and John Borland, "Is a 3D web more than just empty promises?". CNET News.com, May 19, 2005.

[5] P. Corke, *Robotics vision and control. Fundamental algorithms in MATLAB®,* 3rd ed, London, Springer, 2011