Chapter 6

Robot Engraving Services in Industry

Silvia Anton, Florin Daniel Anton and
Mihai Constantinescu

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.69928

Abstract

This chapter presents a software system created to allow to design complex robot trajectories in the cartesian space, used for tasks like milling or engraving with the aid of a robot. The experimental setup was created using an ABB (ASEA Brown Boveri) IRB 140 robot on which was mounted an engraving tool (which was equipped with a high frequency vibrating pin) or a drilling tool (an electric milling machine controlled with I/O signals) and a computer which is executing a CAD application allowing the design of the path that the robot must follow in order to execute the engraving/milling task. The robot job or task is defined by the set of surfaces of the raw object and of the object after being processed, by subtracting these two surfaces the robot task is given by the set of points resulted after the subtraction. The subtraction operation offers a set of points which must be connected in order to obtain the final trajectory, if the task is to engrave a shape then the tool will be normal to the object surface. The algorithm which is generating the surface is presented along with some experimental results.

Keywords: surface extraction, trajectory definition, task modelling, engraving services, industrial robot

1. Introduction

In the last years the manufacturing has changed from mass production to batch production in which a medium or small volume of products are created, the manufacturing process is executed on demand, batches being executed differently based on the client specifications.

Nowadays production lines must be flexible and reconfigurable, changing from a version of a product to another sometimes requires changes in the manufacturing line which higher the costs, because of that there is the trend to use industrial robots which are able to adapt to these changes only by changing the program. The robots are programmed to execute not

only a task, but multiple tasks in order to lower the costs for creating the products, these additional tasks executed by the robots can eliminate other expensive machines such engraving or CNC (computer numerical control) milling machines [1, 2]. In comparison, a robot is much cheaper than a CNC milling machine, but is able to complete milling tasks, with acceptable performances [3, 4].

The main goal of the project that the which is presented in this chapter presents ishas the main goal to developcreate a system which allows to industrial robots to will offer the possibility to design and describe complex surfaces. The system allows to industrial robots to generate complex surfaces which then an industrial robot will follow for executing milling or engraving tasks. The surface will be followed by using programs where the robots will execute procedural motions of the tool, this allows, allowing to processcreate customized, innovative products with a high level of innovation,, which can be used in medicine like personalized prosthesis for correcting orthopaedic affections, creating products based on 3D models defined, in CAD programs or byrecreating objects using reverse engineering using 3D laser scanning, with application in different domainsscans, or using 3D models created in CAD applications, the system having different applications in various fields of activity.

This subject is not approached by the scientific literature, but there are some other related subjects like using robots in milling applications [5–7] or trajectory tracking for robots in milling applications [8, 9]. This chapter is offering examples for robot engraving tasks.

2. The architecture of the system

The chapter is proposing an approach based on a software system allowing the connectivity with vertically articulated industrial robots. The system has three elements (**Figure 1**):

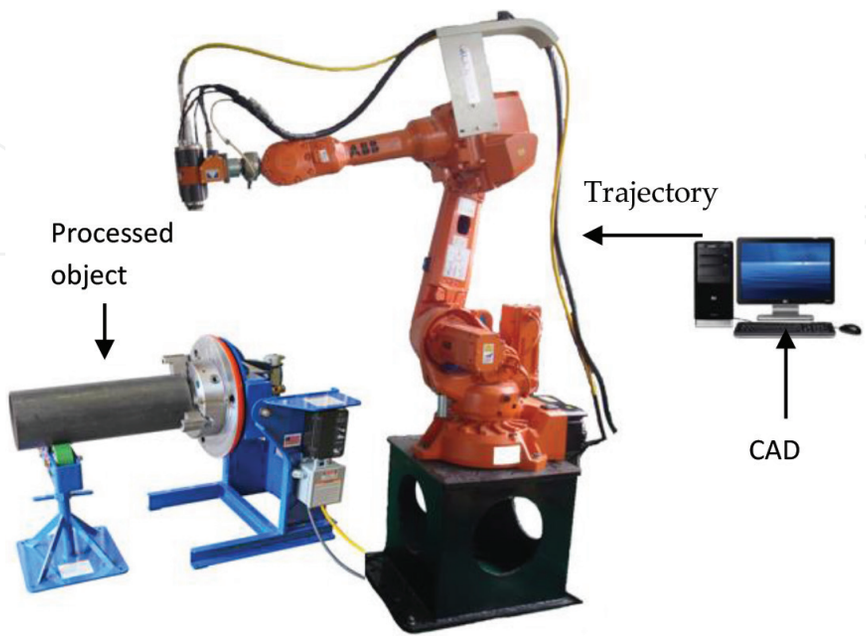

Figure 1. The architecture of the system.

1. A software application which is used to create the surfaces in 3D (CAD), allowing to:

- Import a 3D STL (STereoLitography) model of the raw object (the object before the processing).
- Import an object which describes the 3D surface. This can be done in four ways: by loading a 2D image of the trajectory of the tool; by entering a text which will be used to create an engraving on the processed object; by loading another 3D STL object which defines the robot trajectory in the space; or by loading another 3D object which represents the final product (for example after milling the raw object).
- Create an association between the initial object (without processing) and the trajectory which the robot will follow. This step can be accomplished by overlapping or mapping the object which describes the surface on the initial object: placing the text or the image over the surface of the original object if the robot task will be engraving or low depth milling, placing the 3D STL model which defines the robot trajectory in space over the initial object surface, or by placing the 3D STL model representing the final product inside the 3D model of the raw object.
- Create the trajectory which will be followed by the robot. This is done by associating the points of the surface which was described in a continuous way that can be described in Cartesian space relative to the coordinate system attached to the raw object.

2. The system used to create the robot-object reference.

This element represents a mathematical algorithm and a procedure used to create the association between the coordinate systems of the robot and respectively the coordinate system of the object. This will allow to express the 3D trajectory relative to the robot World coordinate system, the trajectory being defined in the application relative to the object coordinate system, this is done on two types of robots: ABB and OMRON (in RAPID and V_+).

3. Robot applications for trajectory tracking

This element is represented by robot applications used to control the robot to follow the computed trajectory, the applications being created to allow to specify the precision and the speed of tracking and also to specify the tool engagement angle.

3. Robot task modelling

To describe the robot task, the software application is used, the application is generating a trajectory which will be executed by the robot.

In the previous section, we seen that the application can use three different methods to obtain information about the robot task (see **Figure 2**):

- Based on a text input
- Based on a 2D image of the trajectory
- Based on a 3D STL model

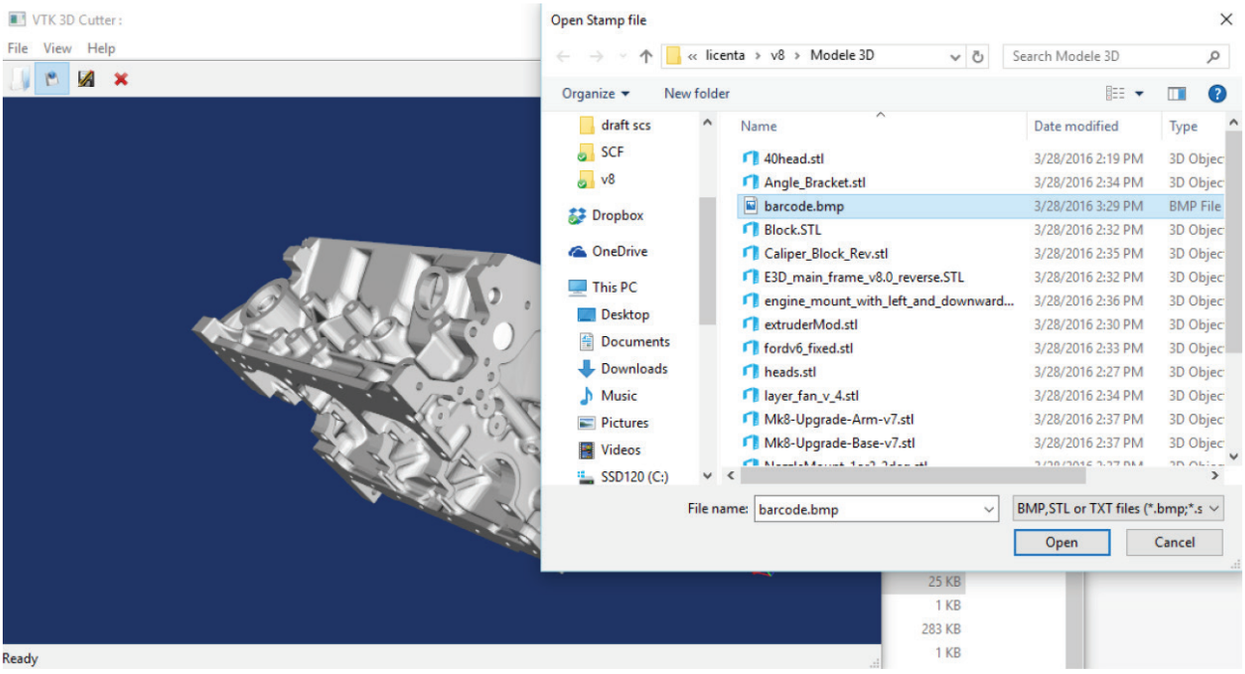

Figure 2. Loading a file in order to define the robot task.

If the user decides to use the first two methods, the input (text or image) is transformed by an image processing algorithm in order to obtain the skeleton. If the input is a text then first it is transformed into an image and then is processed. When a text is introduced into the application also the size of the characters must be introduced (in mm) and, if the task is using a milling device, then the milling depth must be also introduced. In the case of an image also the ratio mm/pixels must be introduced, and the milling depth if the object will be milled. The application is using a skeletonizing algorithm to process the input in order to obtain a simplified trajectory of the robot by thinning the imported path to the size of a line having the width of one pixel.

The skeletonizing algorithm is a process which is reducing the foreground regions in a binary image to a skeleton, which preservers the connectivity and the extent of the original object (region) while discarding most the original foreground pixels. To have a clear picture of how the algorithm is executed, we can imagine that the foreground is made of some uniform material, which is burning slow [10]. If the material is starting to burn simultaneously at all points on the boundary, the fire is advancing on the interior and at the points where is travelling from two different boundaries meets itself and is extinguishing, these points where the fire meets itself and is extinguishing are forming a so-called quench line.

The 'quench line' represents the skeleton, so in respect to this definition, we can state that the thinning operation is producing a sort of skeleton.

Another method of creating the skeleton is to compute the loci of centres of bi-tangent circles which fit entirely within the object (foreground region) which is being considered [11, 12]. **Figure 3** depicts this for a rectangular object.

http://dx.doi.org/10.5772/intechopen.69928

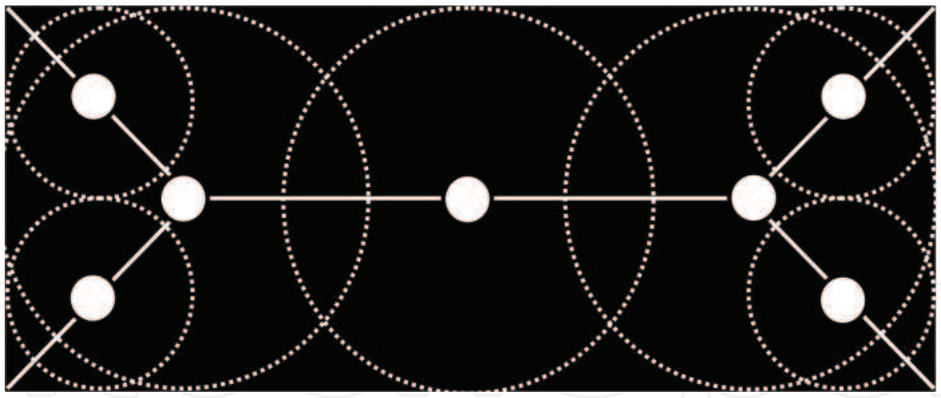

Figure 3. The skeleton created by the centres of bi-tangent circles.

In conclusion, in order to produce the skeleton, we can use two main methods. The first method is to use the thinning algorithm, which successively erodes the pixels from the frontier of object, but at the same time is preserving the end points of the line segments; until no more thinning is possible, the remnant of the object is a shape which is approximating the skeleton. Another possibility is to use an algorithm to process the image in order to obtain the distance transform. After obtaining the distance transform, the skeleton can be generated by connecting the singularities in the distance transform (i.e. discontinuities of the curvature or creases).

The majority of the actual algorithms used for skeletonization are producing skeletons with gaps; these skeletons with discontinuities cannot be used in applications where the shape must be described. This is happening because the homotopy is not preserved and important features like end points or junction points are lost. On the other hand, the thinning algorithm is generating connected skeletons, the 'quench line' being one pixel thick [13].

The thinning algorithm is making part of the class of morphological operators and is used to remove selected foreground pixels from binary images, it is similar with erosion or opening [14, 15], which are members of the same class. Thinning can be used for different applications, but is, in particular, useful for implementing the skeletonizing algorithm. For skeletonizing, thinning is used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning is normally used only for processing binary images, and produces another binary image as an output. Being a morphological operator, the operation of the thinning algorithm is based on a structuring element.

Thinning is related to the hit-and-miss transform and can be expressed based on it, for example, the thinning operation applied on an image I using a structuring element S is given by:

$$thin(I, S) = I - hit\_and\_miss(I, S) \tag{1}$$

where the operator of subtraction is given by: $X - Y = X \cap NOTY$.

The algorithm of thinning is applied by successively placing the centre of the structuring element over the pixels in the processed binary image (the structuring element is covering

the entire image, from left to right and from top to bottom). For each position, the structuring element is compared with the underlying image pixels. If the structuring element matches exactly the pixels from the foreground image, then the corresponding pixel underneath the origin of the structuring element is set to background (zero), otherwise it is left unchanged. In order to have effect, the structuring element should have the origin pixel (the pixel from the centre of the structuring element) set to one or black.

Selecting the structuring element determines the conditions which should be satisfied when a foreground pixel will be set to background, and also, it determines the application for the thinning operation.

Here, we described how the image is modified after a single pass of a thinning operation; in fact, the operator is applied repeatedly until convergence (until it causes no further changes to the image). In other applications (for example, pruning), the operation is only applied for a limited number of iterations, in order to remove only few pixels.

The most common use for thinning, is to reduce the thresholded output of an edge detector like the Sobel operator, to one pixel tick lines, but in the same time preserving the length of those lines (in this way, the pixels at the ends of the lines are not affected). A simple algorithm for doing this is the following:

1. Select all pixels on the boundaries (pixels which have at least one neighbour on the background) of a foreground region (i.e. objects on a segmented image).
2. From the pixels previously selected, delete any pixel that has more than one foreground neighbour, as long as this operation is not disconnecting the region containing the pixel (i.e. split the region into two).
3. Iterate until convergence.

The procedure presented above will erode the edges of the objects in the image until no pixel can be removed (i.e. until convergence), but is not affecting the pixels which are ending the lines.

The same result can be obtained using the operator of morphological thinning which is applied until no change is produced in the image. The structuring elements which can be used are presented in **Figure 4**. The structuring elements are applied successively with all their 90° rotations (totalling 4 × 2 = 8 structuring elements).

This operation is executed in order to determine the octagonal skeleton of a binarized object, the skeleton being composed of the points positioned in the centres of octagons, which are tangent on the edges of the object in at least two points and fit entirely inside the object. Using this method to obtain the skeleton generates a result that is a connected skeleton having one pixel tick lines.

At each loop in the algorithm, the binary object is processed first by the structuring element from the left and then by the one from the right, and after that, using the other six 90° rotations of the two structuring elements.

http://dx.doi.org/10.5772/intechopen.69928

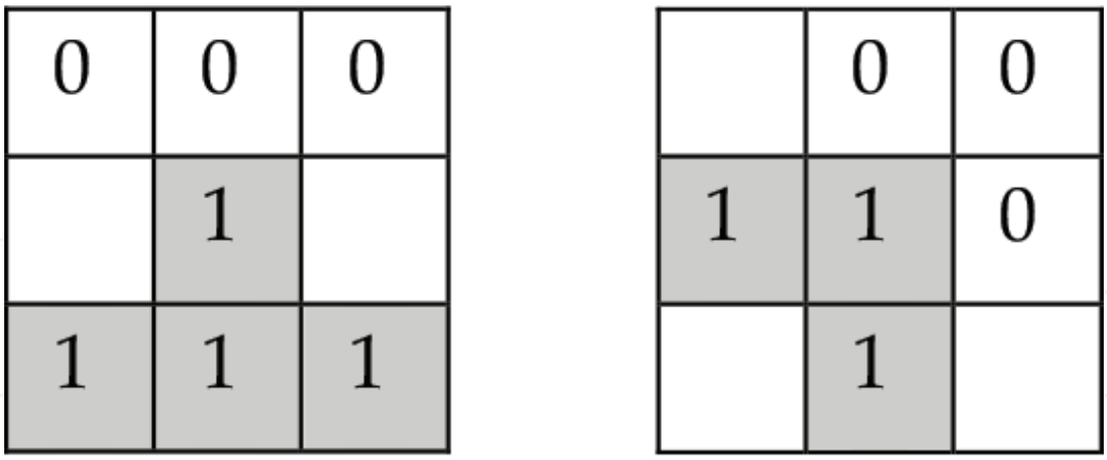

Figure 4. Structuring elements for computing the skeleton (using thinning).

The algorithm is executed in a loop until convergence [16] (until none of the eight structuring elements produces changes in the image), and the object is eroded until it remains only a set connected lines having the width of one pixel.

The thinning operation is very sensitive on perturbations, because of that the irregularities in the edges of the object will generate spurious spurs in the skeleton and this can interfere with shape recognition processes which are using the skeleton properties. In such situations, a pruning operation must be executed in order to delete spurs having a certain length or less. This method is not always efficient because small perturbations in the edge of the object can generate large spurs in the skeleton [17, 18]. Pruning is similar with thinning; the operation is based on the structuring elements presented in **Figure 5**.

In each loop, each structuring element is applied with all its 90° rotations. In contrast with thinning, the pruning algorithm is executed for only a small number of loops in order to

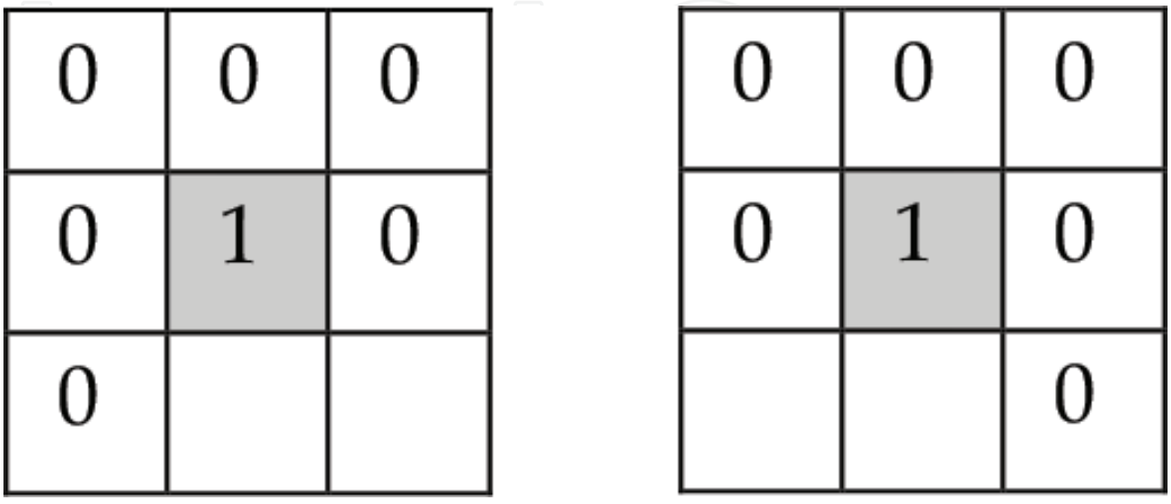

Figure 5. The structuring elements used in pruning.

remove the spurs, depending on their length. If executed until convergence, the pruning algorithm will delete all pixels except those which form closed loops.

When the image processing is done, the image which results (containing the text) can be mapped on the object which will be processed by the robot. In **Figure 6** an example is presented, the first image represents the text introduced initially, then the skeleton of the text is presented after applying the pruning algorithm, and the third image represents the same input but using a dotted font used when a continuous path is not required (for example when engraving a serial number.

The skeletonizing algorithm was also used to process input images which are describing more complex robot trajectories than text input. In **Figure 7** is presented a logo which will be engraved, the second image presents the image after has been processed by the skeletonizing and pruning algorithm, the resulted shape is composed by one pixel tick lines.

If the task is defied by using a 3D object imported into the application, the object is aligned with the initial object and placed on its surface, this is done by executing the algorithm described in **Figure 8** which is using the Iterative Closest Point (ICP) algorithm [19]. The source object represents the object which is used to define the trajectory and the destination object is the object which will be processed by the robot. **Figure 9** presents a 2D code (in the left of the figure) and in the right the code placed on the object after the execution of the ICP algorithm.

When the alignment between the two objects has been executed the ICP algorithm is applied, and then using the VTK (Visualization Toolkit) Poly Data algorithm the surface points are extracted. This algorithm which executes is filtering the pixels of based on the intersection of the two objects [20]. VTK Poly Data algorithm is implemented in python; in **Figure 10**, we can see how the algorithm is integrated with the other objects in python in order to extract the points. In **Figure 11**, we can see an example of subtraction.

Figure 6. Task definition using a text.

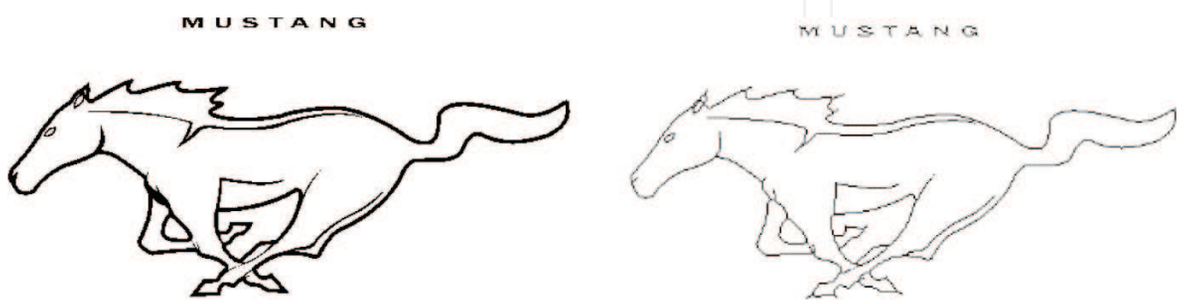

Figure 7. A logo processed for engraving.

http://dx.doi.org/10.5772/intechopen.69928

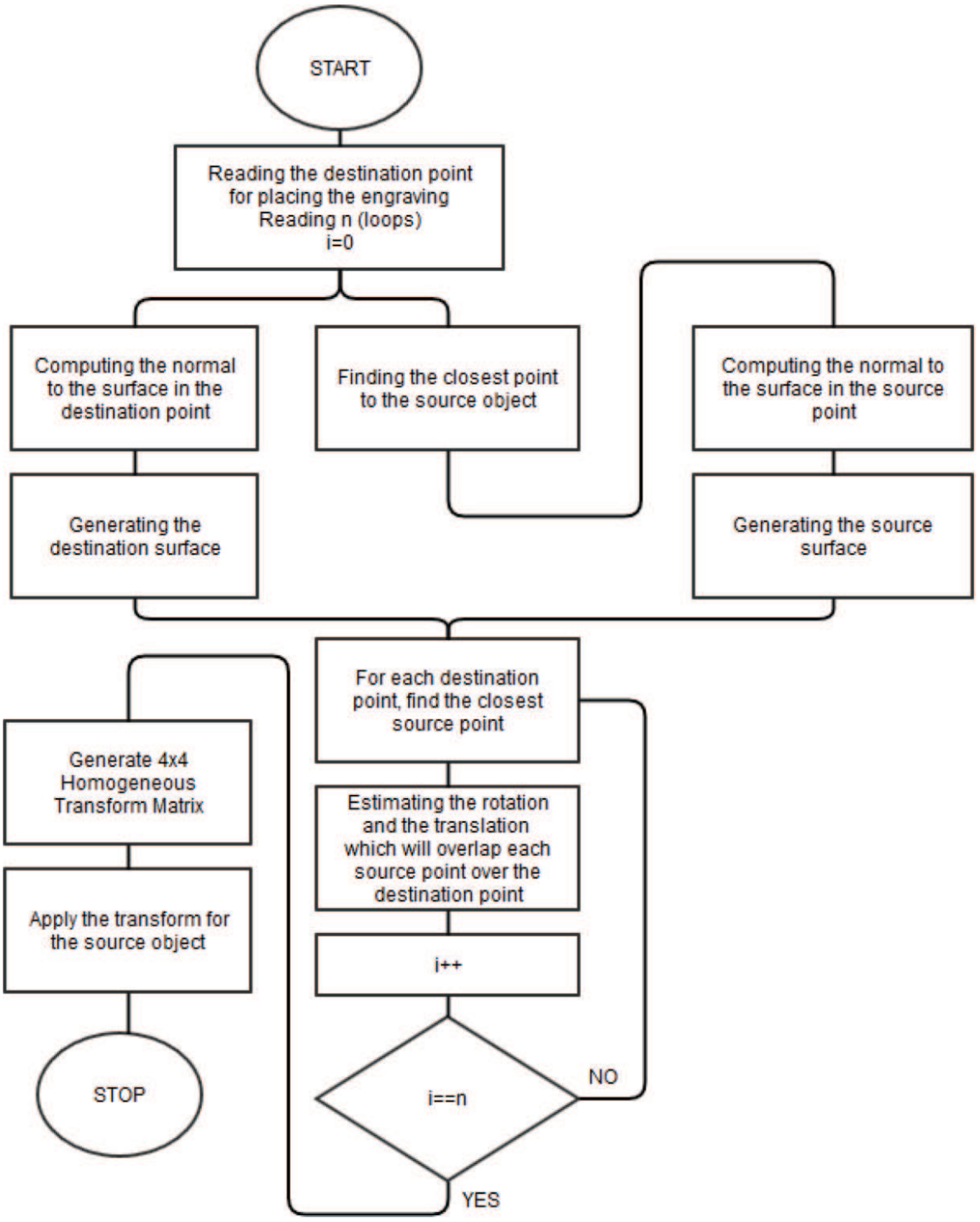

Figure 8. Algorithm for placing the engraving on the object.

The algorithm is allowing to view the result after the task has been executed (the result of the engraving), and also is extracting the points which should be accessed by the robot in order to engrave the object, this is done only when the engraving is based on points and not on continuous paths.

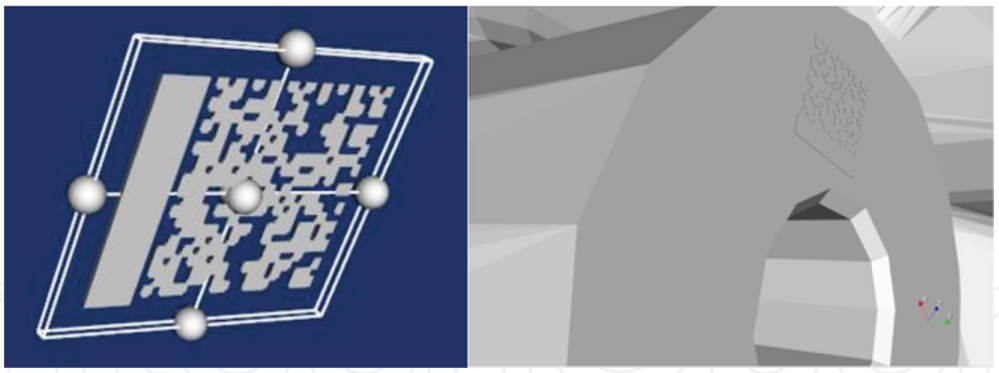

Figure 9. Using an STL model to define the robot task.

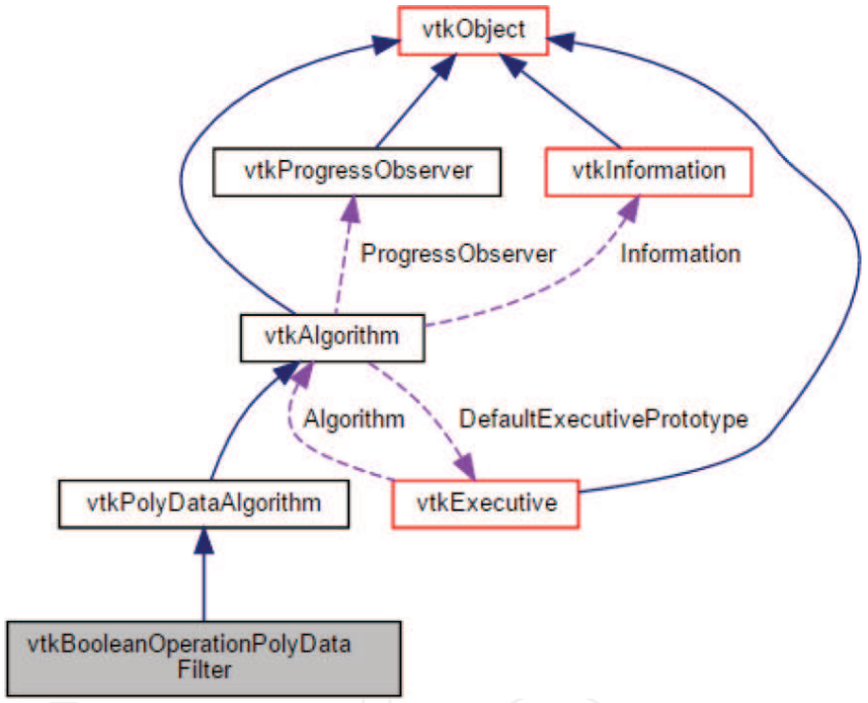

Figure 10. The integration of VTK Poly Data algorithm.

The points which are defining the trajectory are specified relative to the object coordinate system as follows:

a. In the case of using a text or an image to describe the engraving, then the skeleton is followed and the trajectory is defined by the sequence of the skeleton pixels.

b. If the case of using a dotted text, then the coordinates of the points describing the task are selected and sorted based on the distance from the coordinate system and then based on the distance from the last accessed point on the trajectory.

c. If the case of using an STL object in order to describe the trajectory, the task is described by a set of points resulted after executing the VTK Poly Data Algorithm, these points are then sorted like in case b).

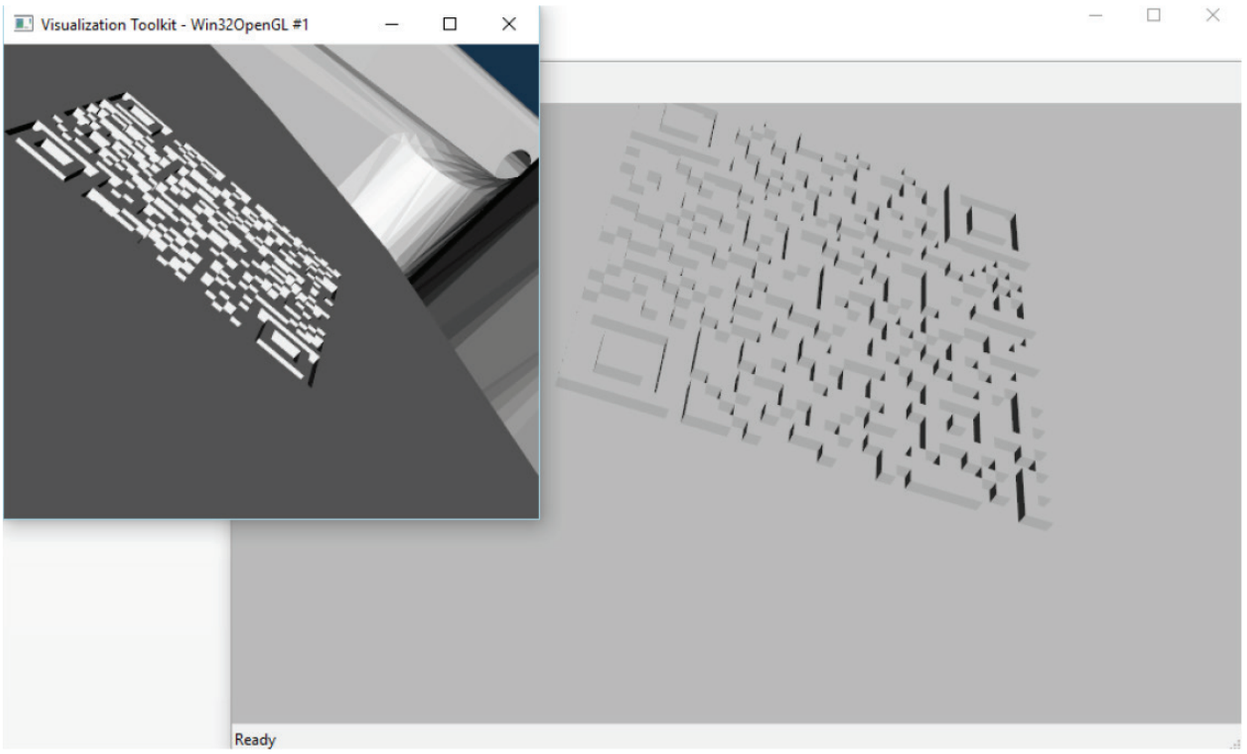

Figure 11. Subtracting a 2D code from an object surface based on VTK Poly Data algorithm.

The points which are describing the trajectory which now are placed in the order that the robot must follow are saved into a text file. This file is describing the robot task, and is sent over an ethernet connection to the robot which is using it to execute the engraving based on procedural motions.

4. Trajectory generation and tracking

In section three of the chapter we seen that the system is extracting the trajectory points which are describing the task which the robot will execute. These points are defined relative to a coordinate system which is known to the application but not to the robot system, in order that the robot to be able to execute the task, the coordinate system must be known by the robot and this is done by a robot-application calibration.

The calibration must be executed before the trajectory points are extracted and sent to the robot, and this is done by defining the object coordinate system in the application based on features which are accessible to the robot.

Figure 12 presents how the coordinate system is defined on the application. The coordinate system is defined using three points on which the robot has access:

a. The coordinate system origin is represented by point A

b. The OX axis is given by AB, where the direction is given by following the line from A to point B

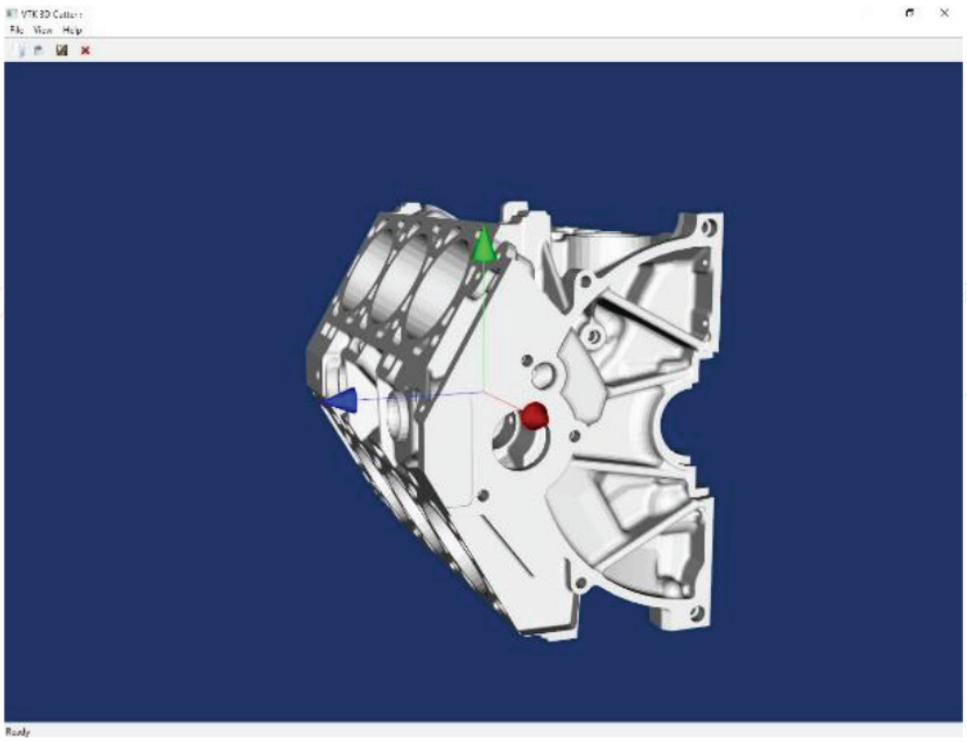

Figure 12. Placing the coordinate system on the object.

c. C is placed on the object surface, if a normal to the direction AB is generated through C, then this normal represents the OY axis. The last axis OZ, is generated by applying the right-hand rule.

Using the three points (A, B and C) trained relative to the robot World coordinate system, a frame attached to the object is defined [21] executing the command FRAME(A,B,C,A).

The resulted coordinate system is used to define the coordinates of the trajectory points in space.

The defined trajectory is followed by the robot using two methods:

a. In the case that the robot trajectory is composed by a set of points which are describing a continous path (based on a text or image) the robot enters in contact with the object using the tool and then is executing the trajectory by moving to each defined point. The tool is departed only when the next point on the trajectory is placed on a distange greater than a defined tolerance (when the trajectory is compose by multiple closed loops or discontinued lines).

b. In the case that the robot trajectory is composed by a set of points which are placed on distances greater than the defined tolerance (for example when the robot task is to engrave a dotted text, a dot code or is based on 3D STL objects), for each point on the trajectory the robot places the tool in contact with the object and then the tool is departed and is placed on the object on the next point of the trajectory.

http://dx.doi.org/10.5772/intechopen.69928

The execution time for the engraving operation is strictly related to the optimum path which will be followed by the engraving tool. A perfect trajectory will generate small holes distributed on a surface according to Ref. [22]. On the other hand, the best engraving time will be achieved if the holes are as small as possible (in terms of height).

The strategy for engraving is the drilling strategy [23]. Here, the engraving is based on drilling (in our case pointing) holes on a surface at different distances. This approach is particularly appropriated for engraving 2D codes.

In order to follow the trajectory, the set of points which are describing the 2D code must be placed on the trajectory in such way that, for example, if the tool is in the point, the next point on the trajectory will be the closest point in the code. Due to this constraint, the points on the trajectory will be arranged in ascending order based on their coordinates. A solution for finding the shortest trajectory is the algorithm Travelling Salesman Problem (TSP), this algorithm can be used for complicated engravings, in our case, for engraving a 2D code, the simplest solution is to follow each row of the 2D code.

To rearrange the points, we start from an arbitrary point (typically, this is a corner), then the closest point is added, the next point is selected from the points which have not been selected and which is the closest point relative to the last point added. The algorithm is iterated until all the points are selected.

By sorting the points based on distances does not create the trajectory, but is a useful step for determining all neighbour points for a trajectory point if we need to modify the trajectory in order to reach a point, which has been skipped when the closest method was used in order to select the points.

An example of a 2D code and the way it is engraved is given in **Figure 13**.

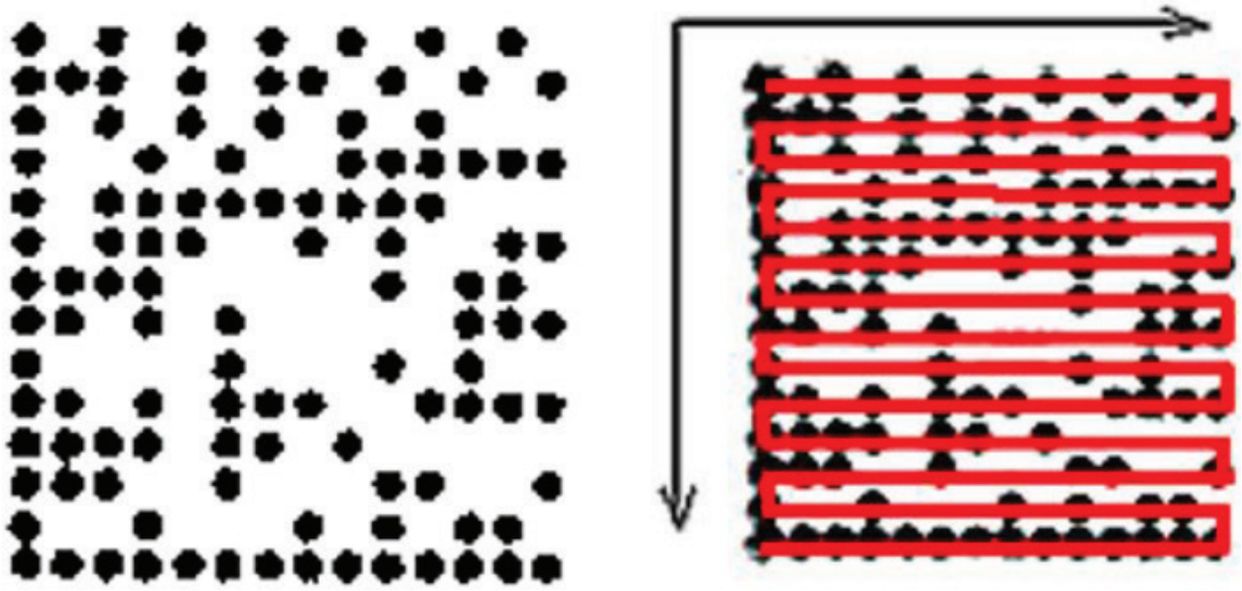

Figure 13. A 2D code and the trajectory used for engraving.

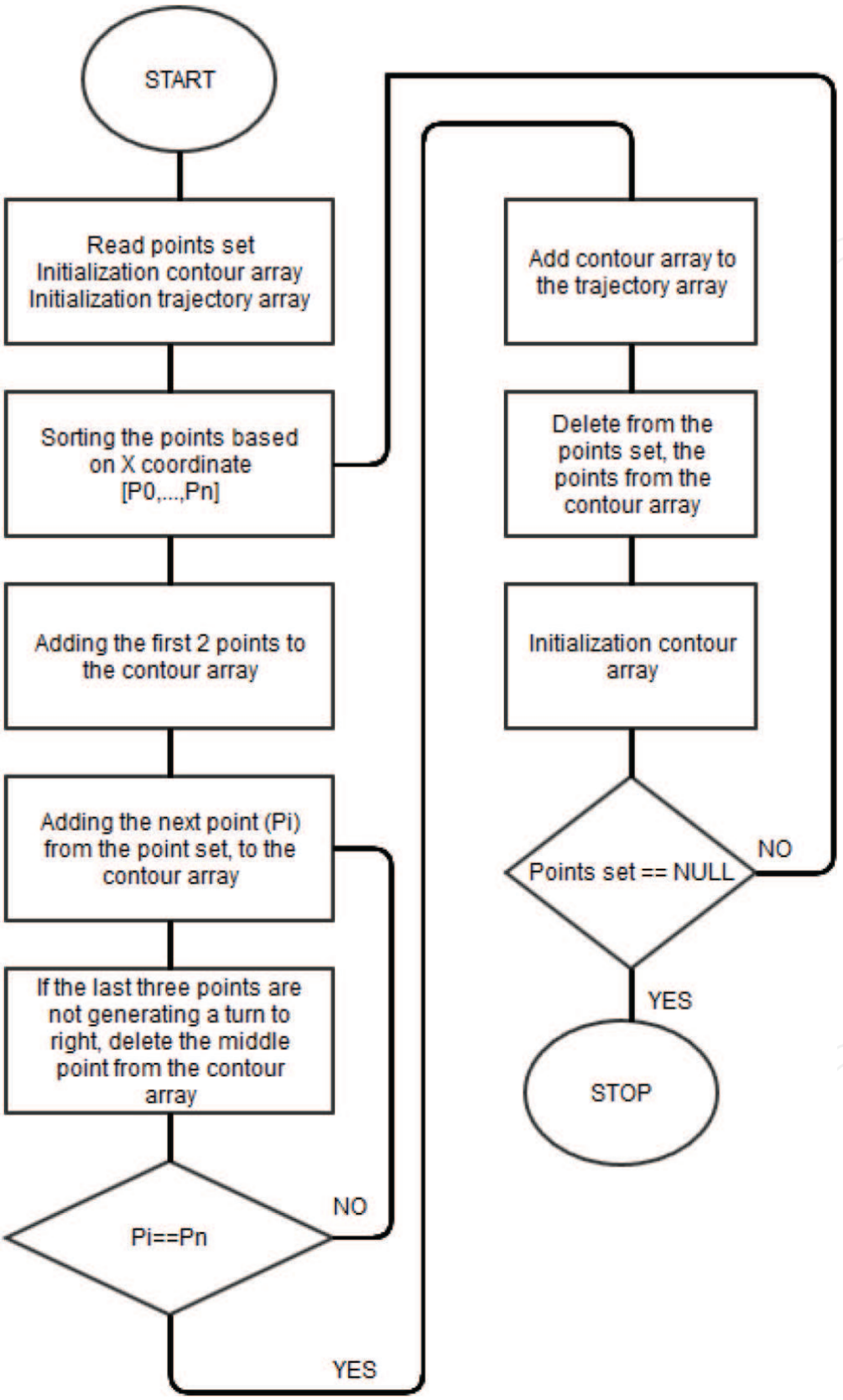

Figure 14. Trajectory generation algorithm.

If the engraving is not a 2D code, we can use the same algorithm; some other algorithms which use the same principle are Graham Scan and Andrew monotone chain.

In the case of an engraving, which represents a closed surface, we can use the following algorithm [24] in order to compute the trajectory:

1. A point P_0 is selected based on its position (right, bottom).
2. The points of the engraving are sorted based of the X coordinate, obtaining a sorted array $[P_0, P_1, \ldots, P_n]$.
3. The first two points are added to the list of contour points.
4. A third point is added (in this case P_3).
5. If the last three points are not creating a turn to right, then the point in the middle (P_2) will be removed from the list. For this, we are testing if the equation$(x_2 - x_1) * (y_3 - y_1) - (y_2 - y_1) * (x_3 - x_1) < 0$ is satisfied (that means that P_2 is in the right side of P_1P_3).
6. Another point is added and the algorithm jumps to step 5 until all points are processed.

The resulted list is the exterior contour, the next step is to find the closest point to P_0 which is not part of the contour points list, this list is extracted from the engraving set of points and the algorithm is applied until no more points are left. In this way, we obtain a set of contours (which are forming the trajectory) which if they are followed by the engraving tool, the result will be the designed engraving. In **Figure 14**, the algorithm is presented.

5. Conclusion

The presented system was designed to allow the implementation of engraving services in production systems to develop new products, on demand, with a higher level of customization. The system can be considered when replacing an engraving or CNC milling machine if the task is not too complex or the execution time is not essential, so depending on the complexity of the task and the execution time the system can be a viable solution.

The system had two implementations based on two types of robots: a Viper s650 produced by Omron (former ADEPT) and an ABB IRB140 robot (see **Figure 15**). The engraving tool which was used was a Parkside hobby engraving machine equipped with a high frequency oscillating vanadium pin.

Based on the executed tests, we observed that the generated engravings are accurate if they are generated using a discrete set of points, the results are also influenced by the quality of the engraving tool which is wobbling if the amplitude of the vibrations is higher or if the trajectory is followed with a higher speed. The possible applications can be in tasks requiring short dot engravings (for example, in automotive industry).

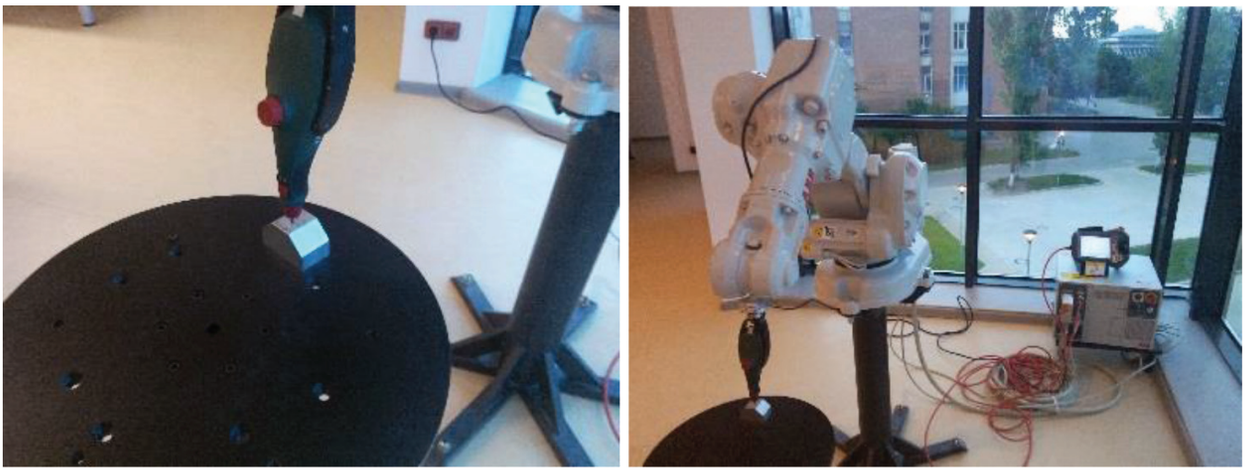

Figure 15. The implementation setup.

The integration of the system is easy and is done only by defining the robot task, adding the tool, and changing the robot programs for PC communication and task execution.

Acknowledgements

This work has been funded by University Politehnica of Bucharest, through the 'Excellence Research Grants' Program, UPB – GEX. Identifier: UPB – EXCELENȚĂ – 2016 Generarea Suprafetelor Complexe in Robotica (GSCR), 2/26.09.2016.

Author details

Silvia Anton, Florin Daniel Anton* and Mihai Constantinescu

*Address all correspondence to: anton@cimr.pub.ro

Department of Automation and Industrial Informatics, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania

References

[1] Brell-Cokcan S, Braumann J. A new parametric design tool for robot milling. In: ACADIA; ISBN: 978-1-4507-3471-4. 2010. pp. 357-363

[2] Pandremenos J, Doukas C, Stavropoulos P, Chryssolouris G. Machining with robots. In: Proceedings of 7th International Conference on Digital Enterprise Technology (DET2011); 2011; University of Patras; Laboratory for Manufacturing Systems and Automation; Athens

[3] Atmosudiro A, Keinert M, Karim A, Lechler A, Verl A, Csizar A. Productivity increase through joint space path planning for robot machining. In: UKSim-AMSS 8th European Modelling Symposium; 21-23 October 2014. Pisa, Italy: IEEE Computer Society; 2014. pp. 257-262

[4] Coelho RT, Rodella HHT, Martins VF, Barba RJ. An investigation into the use of industrial robots for machining soft and low density materials with HSM technique. ABCM. 2011;**XXIII**(3):343-350

[5] Klimchik A, Ambiehl A, Garnier S, Furet B, Pashkevich A. Efficiency evaluation of robots in machining applications using industrial performance measure. Robotics and Computer-Integrated Manufacturing. 2017;**48**:12-29

[6] Taner Tunc L, Gu Y, Burke G. Effects of minimal quantity lubrication (MQL) on surface integrity in robotic milling of austenitic stainless steel. Procedia CIRP. 2016;**45**:215-218

[7] Vosniakos GC, Matsas E. Improving feasibility of robotic milling through robot placement optimisation. Robotics and Computer-Integrated Manufacturing. 2010;**26**(5):517-525

[8] Maciej P, Grzegorz K, Konrad G, Grzegorz G, Konrad K, Janusz O. Trajectory tracking controller of the hybrid robot for milling. Mechatronics. 2016;**37**:100-111

[9] Gracia L, Garelli F, Sala A. Integrated sliding-mode algorithms in robot tracking applications. Robotics and Computer-Integrated Manufacturing. 2013;**29**(1):53-62

[10] Jain A. Fundamentals of Digital Image Processing. Prentice-Hall; USA; 1989. Chapter 9

[11] Fisher R, Perkins S, Walker A, Wolfart E. Image Processing Learning Resources [Internet]. Available from: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm. 2004

[12] Ballard D, Brown C. Computer Vision. Prentice-Hall; USA; 1982. Chapter 8

[13] Serra J. Image Analysis and Mathematical Morphology. Academic press; Amsterdam; Netherlands; 1992

[14] Davies E. Machine Vision: Theory, Algorithms and Practicalities. Academic Press; Amsterdam; Netherlands; 1990. pp. 149-161

[15] Haralick R, Shapiro L. Computer and Robot Vision. Vol. 1. Addison-Wesley Publishing Company; Boston; USA; 1992. pp. 168-173. Chapter 5

[16] Soille P. Morphological Image Analysis: Principle and Applications. Springer; Berlin; Germany; 1999

[17] Gonzalez R, Woods R. Digital Image Processing. Addison-Wesley Publishing Company; Boston; USA; 1992. pp. 518-548

[18] Vernon D. Machine Vision. Prentice-Hall; USA; 1991. Chapter 4

[19] Pomerleau F, Colas F, Siegwart R. A review of point cloud registration algorithms for mobile robotics. Foundations and Trends in Robotics. 2015;**4**(1):1-104

[20] Quammen C, Weigle C, Taylor II RM. Boolean operations on surfaces in VTK without external libraries. Insight Journal. 2011;**53**:1-12. DOI: http://hdl.handle.net/10380/3262

[21] Adept Technology Inc, editor. A dept V+ Reference Guide. Version 14.0, Part Number 00964-03000, Rev. B ed. San Jose, CA: Technical Publications; 2001

[22] Warkentin A, Hoskins P, Ismail F, Bedi S. Computer Aided 5-axis Machining, Computer Aided Design, Engineering and Manufacturing: System Techniques and Applications. CRC Press; London; United Kingdom; 2001. Chapter 3

[23] Mladenovic GM, Tanovic LM, Ehmann KF. Tool path generation for milling of free form surfaces with feedrate scheduling. FME Transactions. 2015;**43**(1):9-15

[24] Constantinescu M. Defining 3D robot trajectories for complex applications [dissertation]. Bucharest: UPB; 2016